



Title	2次元ビン・パッキング問題と真円度問題に関する研究
Author(s)	榎原, 博之
Citation	大阪大学, 1987, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/1408
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

2次元ビン・パッキング問題と 真円度問題に関する研究

昭和 61 年 12 月

榎 原 博 之

内容梗概

本論文は、著者が大阪大学大学院工学研究科通信工学専攻在学中に行った研究のうち、計算幾何学の分野に属する2次元ビン・パッキング問題と真円度問題に関する研究をまとめたものである。

本論文では、まず、幾何学的最適化問題の1つである2次元ビン・パッキング問題を取り上げ、この問題に制約を加えた場合の厳密解法と近似解法についての研究を、ついで、計算幾何学の近接問題の中心課題の1つであるボロノイ図を応用して、真円度を求める多項式時間の厳密解法についての研究を述べている。

第2章、第3章では、ピースに制約を加えた2次元ビン・パッキング問題について考察している。第2章では、ピースに制約を加えた2次元ビン・パッキング問題に対する3つの厳密解法を提案し、その最適性について論じ、少なくともピースの種類が 4×1 、 1×4 、 2×2 の部分矩形に限定した場合までは、線形時間で最適解が得られることを明らかにしている。

第3章では、ピースに制約を加えた2次元ビン・パッキング問題に対する3つの近似解法を提案し、その良さを評価している。まず、ピースの種類を 3×2 の部分矩形に限定した場合の近似解法を提案し、その最悪の場合の近似度を理論的に解析している。次に、ピースの種類を $w \times 1$ および $w \times 2$ ($w \geq 5$) の部分矩形に限定した場合の近似解法を提案し、その良さを計算機実験によって評価している。

第4章では、ボロノイ図を利用することによって、機械部品の精度として重要な幾何公差の1つである真円度を求める多項式時間の厳密解法が存在することを指摘し、その最適性と時間計算量を明らかにしている。

関連発表論文

I. 電子通信学会論文誌

- (1) 榎原, 中野, 中西: "ピースに制約を加えた2次元ビン・パッキング問題", 信学論(A), vol.J69-A, no.3, pp.350-358 (Mar. 1986).
- (2) 榎原, 中野, 中西, 村松: "ピースに制約を加えた2次元ビン・パッキング問題に対する近似解法", (ショートノート), 信学論(A), vol.J69-A, no.8, pp.1015-1017 (Aug. 1986).
- (3) 榎原, 中野, 中西, 真田: "ボロノイ図を応用した真円度を求める解法", 信学論(A) (掲載予定).

II. 電子通信学会研究会資料及び全国大会予稿

- (1) 榎原, 中野, 中西: "ピースに制約を加えた2次元ビン・パッキング問題", 信学技報, CAS84-82 (昭59-08).
- (2) 榎原, 中野, 中西: "ピースの種類を3x2の部分矩形に限定した2次元ビン・パッキング問題", 信学会総合全国大会, 44 (昭60-03).
- (3) 榎原, 中野, 中西, 村松: "ピースに制約を加えた2次元ビン・パッキング問題に対する近似解法", 信学技報, CAS85-88 (昭60-10).

III. Proceedings of ISCAS 85

- (1) H.EBARA, H.NAKANO, and Y.NAKANISHI: "Constrained Two-Dimensional Bin Packing Problem", Proceedings of ISCAS 85, pp.965-966 (1985).

目 次

第 1 章	緒論 -----	1
第 2 章	ピースに制約を加えた 2 次元ビン・パッキング問題の 厳密解法 -----	5
2.1	緒言 -----	5
2.2	ピースの高さが一定である場合 -----	7
2.3	ピースの幅が一定である場合 -----	3 4
2.4	ピースの種類が 2×2 の部分矩形である場合 -----	3 7
2.5	結言 -----	4 0
第 3 章	ピースに制約を加えた 2 次元ビン・パッキング問題の 近似解法 -----	4 2
3.1	緒言 -----	4 2
3.2	近似解法の理論的解析 ----- ～ピースの種類が 3×2 の部分矩形である場合	4 3
3.3	近似解法の計算機実験 -----	5 2
3.3.1	ピースの種類が $w \times 1$ および $w \times 2$ の部分矩形である 場合 -----	5 3
3.3.2	計算機実験による評価 -----	5 8
3.4	結言 -----	6 1
第 4 章	ボロノイ図を応用した真円度を求める解法 -----	6 2
4.1	緒言 -----	6 2
4.2	最近点および最遠点のボロノイ図とそれらの構成算法 --	6 3
4.2.1	最近点および最遠点のボロノイ図 -----	6 3
4.2.2	最近点および最遠点のボロノイ図の構成算法 -----	6 5

4.3	真円度を求める解法	68
4.4	結言	73
第5章	結論	74
謝辞		76
参考文献		77

第 1 章 結 論

近年，コンピュータ・サイエンスの中で急速に発展してきた分野の 1 つに計算幾何学 (computational geometry) [2][9][11][12] と呼ばれる分野がある．これは，幾何学 (geometry) に計算の複雑さ (computational complexity) の理論 [1] を導入して，幾何学的な問題を計算機で効率良く処理する算法を開発し，その限界等を理論的に究明する学術分野である．計算幾何学は，VLSI の CAD，パターン認識，コンピュータ・グラフィックス，地理情報処理，ロボティクス，データベース理論等応用分野が非常に広く，これら以外の分野にも広く応用される可能性を秘めている．

これまでの幾何学的な問題に対する算法は，人間の直観（パターン認識）によるものが多く，問題の規模が小さければすぐに解を求めることができるが，大規模な問題をしかも計算機を使って解こうとする場合うまくいかないことが多かった．しかし，コンピュータ・サイエンスの分野で算法やデータ構造の研究 [1][15] が急速に進歩し，これらの算法やデータ構造を利用することにより，あるいは，これらの算法やデータ構造の考え方を基に新たな算法やデータ構造を開発することにより，幾何学的な情報を計算機で効率的に処理できるようになってきている．

計算幾何学の代表的な問題は大きく次のように分類される [11]．

(1) 凸包 (convexity) 問題

与えられた幾何学的対象物（たとえば，点など）の集合に対してその集合の凸包 (convex hull) を求める問題．

(2) 重なり (intersection) 問題

与えられた幾何学的対象物（たとえば，線分など）の交差判定をしたり共通部分を求めたりする問題．

(3) 幾何学的探索 (geometric searching) 問題

ある種の幾何学的な図形（たとえば，地図など）に対してある点を含む領域を決定するとか，ある幾何学的な図形に対してその内部に含まれる幾何学的対象物（たとえば，点など）を列挙するなどという探索問題．

（４） 近接（proximity）問題

ある幾何学的対象物（たとえば，点など）間の近接関係等を調べる問題．この問題にはしばしば計算幾何学の中心課題の１つであるボロノイ図が使われる．

（５） 幾何学的最適化（geometric optimization）問題

与えられた幾何学的対象物に対する各種の最適化問題．

なお，計算幾何学の問題に使用される主な算法には，平面走査法（plane sweep method），分割統治法（divide-and-conquer method），幾何学的変換法（geometric transformation method），バケット法，フィルタリング法等があり，データ構造には，平衡二分木（balanced binary tree），セグメント木，ヒープ探索木（priority search tree）等がある．

本論文では，幾何学的最適化問題に属する２次元ビン・パッキング問題（2-Dimensional Bin Packing Problem）^{〔5〕}と真円度問題^{〔16〕}を取り上げている．２次元ビン・パッキング問題については，この問題に制約を加えた場合の厳密解法と近似解法について考察している．２次元ビン・パッキング問題は，NP 完全^{〔7〕}，あるいはそれ以上のクラスに属する扱いにくい問題であり，問題に制約を加えない限り多項式時間で最適解を得ることは不可能であると考えられている．しかし，もしピースの種類に制約を加えるならば，その扱いにくさ（intractability）^{〔7〕}が変わるものと考えられる．そこで，この問題の扱いにくさがピースの種類によってどのように変わるかを系統的に追究しやすくするために，ビンの幅，ピースの幅，

高さをすべて整数値とし、かつピースの種類を $w \times h$ の部分矩形（幅 w 以下、高さ h 以下の矩形の集合）に限定する．そして、 w および h を逐次増加させて、時間計算量がピース数 n の多項式時間、とくに $O(n)$ 回の BL（ボトム・レフト）パッキング^[3]での厳密解法および極めて良い近似解法の存在する w および h を見い出すことを試みている．

第2章では、ピースの種類を $w \times h$ の部分矩形に限定し、 w および h を逐次増加させて、時間計算量がピース数 n の線形時間である厳密解法の存在する w および h を見い出すことを試み、少なくともピースの種類が 4×1 ， 1×4 ， 2×2 の部分矩形に限定した場合までは、線形時間で最適解が得られる解法が考えられることを明らかにしている．

第3章では、ピースの種類を $w \times h$ の部分矩形としても、 w および h の値が第2章で見い出した厳密解法で考えられる整数値を越えると、もはや時間計算量がピース数 n の線形時間となる厳密解法は期待できそうにないとの判断のもとに、近似解法について考察している．このような場合でも、第2章で見い出した厳密解法の考え方や解法そのものを活用することにより、良い近似解法を構成することができる可能性があると思われる．第3章では、まず、近似解法の理論的解析として、ピースの種類を 3×2 の部分矩形に限定した場合の近似解法を提案し、その最悪の場合の近似度を示している．この近似解法は、第2章のピースの種類が 4×1 の部分矩形である場合の厳密解法の考え方を基にしたものであり、線形時間で最悪の場合でもたかだか高さが1しか悪くならない近似解が得られることを明らかにしている．次に、ピースの種類を $w \times 1$ および $w \times 2$ （ $w \geq 5$ ）の部分矩形に限定した場合の近似解法を提案し、その良さを既存の近似解法であるレベル・アルゴリズム^[6]との比較実験を通して評価している．これらの近似解法は、第2章のピースの種類

が 4×1 の部分矩形である場合の厳密解法を活用したものであり、実験結果から計算時間も妥当であり、性能の点でも良い近似解法であると考えられることを明らかにしている。

真円度問題については、近接問題の中心課題の1つであるボロノイ図^{[8][13]}を利用することによって、多項式時間で真円度^[16]を求めることができることを明らかにしている。真円度を求める解法に関しては、今までのところ近似解法に過ぎないとか、厳密解法であっても時間計算量に難点があるものであって、時間計算量が多項式時間である厳密解法は知られていなかった。

第4章では、ボロノイ図を利用することによって、機械部品の精度として重要な幾何公差の1つである真円度を求める多項式時間の厳密解法が存在することを指摘し、その最適性と時間計算量を明らかにしている。この解法は、最近点のボロノイ図（勢力圏図）と最遠点のボロノイ図（非勢力圏図）の結び（Union）をとるという考えに基づいたものである。

第2章 ピースに制約を加えた 2次元ビン・パッキング 問題の厳密解法

2.1 緒言

本章では、ピースに制約を加えた2次元ビン・パッキング問題に対する3つの厳密解法を提案し、その最適性について議論する。

2次元ビン・パッキング問題^[5]とは、

“幅 W で、高さが無限大のオープン・エンドの矩形のビン B と、幅が w_i 、高さが h_i の n 個の矩形のピース p_i ($1 \leq i \leq n$) が与えられたとき (ただし、 $w_i \leq W$)、すべてのピース p_i を互いに重なることなく、高さが最小となるように詰める問題”である (図2.1)。なお、 n 個のピースは、リスト L の中に添字の順に入っているものとし、また、ピースの回転を許さないものとする。この問題は、マルチ・プロセッサのジョブ割当てを始め、型紙のレイアウト等、応用分野も広く、その解法についての研究が活発に進められてきており、これまでにいくつかの近似解法が提案されてきている^{[3][6][14]}。

2次元ビン・パッキング問題は、1次元ビン・パッキング問題^{[5][10]}がNP完全な問題である^[7]ことから考え、NP完全^[7]、あるいはそれ以上のクラスに属する問題であると考えられており、一般には扱いにくい問題である。しかし、もしピースの種類が制約されれば、扱いにくさ (intractability)^[7]が変わるものと考えられる。本章と次章は、ピースの種類によって扱いにくさがどのようになるかを明らかにすることを目的にしたものであって、このことを系統的に追究しやすくするために、ビンの幅、ピースの幅、高さをすべて整数値とし、ピースの種類を $w \times h$ の部分矩形に限定する。本章では、 w および h を逐次増加させて、時間計算量がピース

数 n の多項式時間，とくに線形時間となる厳密解法の存在する w および h を見い出すことを試みている．

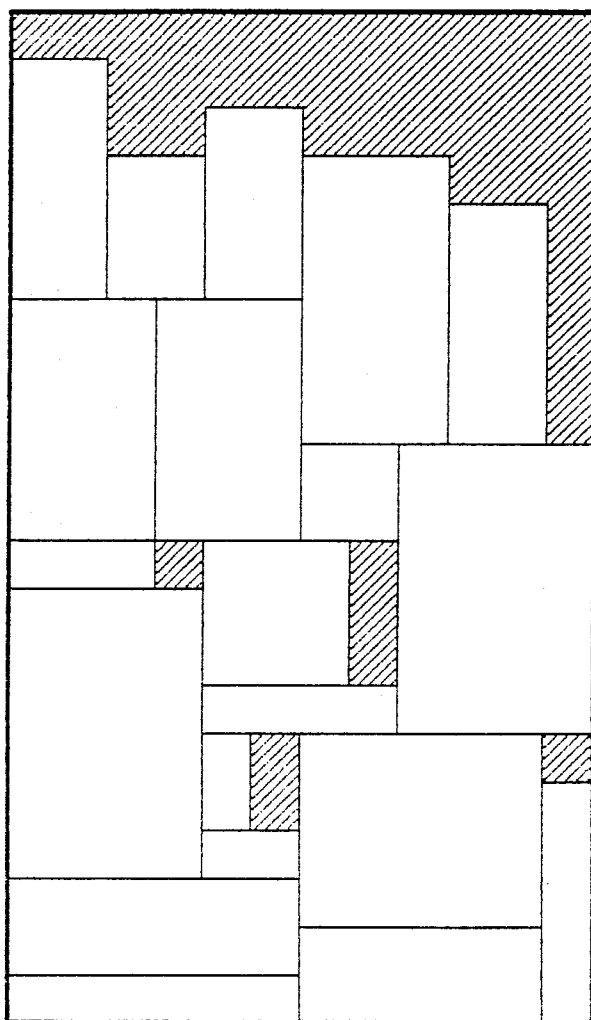


図 2 . 1 2次元ビン・パッキング問題のパッキング例

ここで、本章と次章で使用する特別な用語と記号をまとめて示しておく。

- ・ “BL (ボトム・レフト) パッキング” …… 空き地も考慮して、ピースを詰めることができる最も底に近い位置で、かつその中で最も左の位置にピースを詰めること^[3]。また、後述の解法で使用している “BL で詰める” とは、この規則に従いピースを詰めることである。
- ・ “空き地” …… 周囲をピースまたはビンで囲まれたピースの存在しない領域。なお、右端にできる、上にのみ開いた細長いピースの存在しない領域も空き地と呼ぶこととする。
- ・ “ $i \times j$ の矩形” …… 幅 i ，高さ j の矩形。
- ・ “ $w \times h$ の部分矩形” …… $\{ i \times j \text{ の矩形} \mid 1 \leq i \leq w, 1 \leq j \leq h, i, j \text{ は整数} \}$ 。たとえば、 3×2 の部分矩形とは、 3×2 ， 3×1 ， 2×2 ， 2×1 ， 1×2 ， 1×1 の6種類の矩形の集合である。
- ・ “タイト” …… 空き地がなく、ピースがピッタリと詰まっている状態。
- ・ “レベル” …… ピースを詰める際に基準となる、ビンの底辺に平行な直線上の、ピースの存在することができる領域^[6]。たとえば、高さ1のピースのみを詰めた場合は、高さ1ごとにレベルができる。
- ・ “フリーのピース” …… ビンに詰められていないピース。
- ・ p_{ij} …… 幅 i ，高さ j の矩形のピース。
- ・ n_{ij} …… 幅 i ，高さ j の矩形のピースの個数。

2. 2 ピースの高さが一定である場合

ピースの高さが一定であるという制約を加えると、問題は単に1次元ビン・パッキング問題^{[5][10]}となる。この問題において、ビ

ピースの種類を $w \times 1$ の部分矩形に限定し， w をパラメータとして多項式時間の厳密解法が考えられるかについて考察する．ピースの種類を 4×1 の部分矩形に限定した場合の厳密解法を提案し，次の定理を導いている．

〔定理 2. 1〕 ピースの種類を 4×1 の部分矩形に限定する．このとき，ピースをアルゴリズム 2. 1 で詰めると，必ず最適に詰めることができる．また，このアルゴリズムは $O(n)$ 回の BL パッキングで行うことができる．ただし， W はビンの幅であり， n はピース数である． □

アルゴリズム 2. 1

メイン・ルーチン

```
step1  if   $W = 4$   then  手続き  $W4$  を行う；
step2  if   $W = 5$   then  手続き  $W5$  を行う；
step3  if   $W \geq 6$   then  begin
    3.1  if   $W \bmod 4 = 0$   then
        手続き  $Wm0$  を行う；
    3.2  if   $W \bmod 4 = 1$   then
        手続き  $Wm1$  を行う；
    3.3  if   $W \bmod 4 = 2$   then
        手続き  $Wm2$  を行う；
    3.4  if   $W \bmod 4 = 3$   then
        手続き  $Wm3$  を行う；
    3.5  手続き exchange を行う；
    3.6  タイトに詰まっていないレベルで，
        手続き repack を行う；
    end
step4   $p_{11}$  を BL で詰める；
```

手続きW4

- step1 p_{41} をBLで詰める；
- step2 p_{21} をBLで詰める；
- step3 p_{31} をBLで詰める；

手続きW5

- step1 p_{31} をBLで詰める；
- step2 p_{21} を p_{31} の最高レベルまでBLで詰める；
- step3 p_{41} をBLで詰める；
- step4 残った p_{21} をBLで詰める；

手続きWm0

- step1 p_{41} をBLで詰める；
- step2 p_{21} をBLで詰める；

手続きWm1

- step1 p_{31} を右端に1列に詰める；
- step2 p_{41} を右端の幅5の領域を除いてBLで詰める；
- step3 p_{31} と p_{41} の間にできた幅2の空き地に、タイトな p_{41} の最高レベルまで p_{21} を詰める；
- step4 もし p_{21} がタイトな p_{41} の最高レベルまで詰めることができなければ、以下の操作をタイトな p_{41} の最高レベルまで、あるいは右端に詰まっている p_{31} を上から順に2個取ると、その時点でタイトに詰まっている最高レベルの高さ(H_t)以下になるまで行う；
 - ・タイトでない最低レベルの一番右の p_{41} を取り除き、そこに右端に詰まっている p_{31} を上から順に2個取って詰め、取り除いた p_{41} をstep2の操作に倣って詰め

る；

- step5 もし p_{31} がタイトな p_{41} の最高レベルまで詰めることができず， p_{21} が p_{31} の最高レベルまで詰めることができるならば， p_{31} の最高レベルまで p_{21} を詰める；
- step6 残った p_{21} を BL で詰める；

手続き Wm 2

- step1 p_{41} を BL で詰める；
- step2 右端の幅 2 の空き地に，タイトな p_{41} の最高レベルまで p_{21} を詰める；
- step3 もし p_{21} がタイトな p_{41} の最高レベルまで詰めることができなければ，以下の操作をタイトな p_{41} の最高レベルまで，あるいは p_{31} が足りなくなる（1 個以下になる）まで行う；
- ・タイトでない最低レベルの一番右の p_{41} を取り除き，そこにフリーの p_{31} 2 個を詰め，取り除いた p_{41} を BL で詰める；
- step4 残った p_{21} を BL で詰める；

手続き Wm 3

- step1 p_{31} を右端に 1 列に詰める；
- step2 p_{41} を BL で詰める；
- step3 p_{21} を BL で詰める；

手続き exchange

- step0 この手続きにおいて交換される p_{31} は， W が偶数のときはフリーの p_{31} であり， W が奇数のときは右端に 1 列に詰められた p_{31} である； このピースを p_{ex} とする；

Wが偶数のときは、交換することができるフリーの p_{31} がなくなる（2個未満あるいは4個未満になる）まで、以下の操作を行う； また、Wが奇数のときは、交換するために右端の一番上の p_{31} を取り除くことによって、その時点でタイトに詰まっている最高レベルの高さ（ H_t ）未満になるならば、交換することは行わず、この手続きは終了する；

- step1 同じレベルにある p_{21} 3個を p_{ex} 2個と交換し、交換された p_{21} 3個をBLで詰める； この操作を行えば、step1へ；
- step2 同じレベルにある p_{21} と p_{41} を p_{ex} 2個と交換する；
もし同じレベルに p_{21} がなければ、前もってそのレベルの別の p_{41} と他の同じレベルにある p_{21} 2個を交換しておく； 交換された p_{41} は、もしあれば、同じレベルにある p_{21} 2個と交換し、その後 p_{21} 3個をBLで詰める；
もし同じレベルに p_{21} 2個がなければ、 p_{41} 、 p_{21} をその順にBLで詰める； この操作を行えば、step1へ；
- step3 同じレベルにある p_{41} 3個を p_{ex} 4個と交換し、交換された p_{41} 3個をBLで詰める； この操作を行えば、step1へ；

手続き repack

- step1 残った p_{31} が3個以下の場合；
- p_{41} 、 p_{21} 、 p_{31} の順にBLで詰める；
- step2 残った p_{31} が4個以上の場合；
- 2.1 p_{21} がなく、 p_{41} が2個の場合；
- p_{41} 2個をBLで詰める；

- ・ p_{41} と同じレベルに p_{31} を詰める；
- ・ もし右端に幅 2 の空き地ができるならば，そのレベルの p_{41} 1 個を取り除き，そこに p_{31} 2 個を詰め， p_{41} をその上のレベルに BL で詰める；
- ・ 残りの p_{31} を BL で詰める；

2.2 その他の場合；

- ・ p_{41} ， p_{31} ， p_{21} の順に BL で詰める；

(証明) このアルゴリズムがピースを最適に詰めることを示すためには，ピースができる限りタイトに詰められていることを示さなければならない．もしピースが一番上のレベルを除いてタイトに詰まっているならば，このパッキングは最適である．もし一番上のレベル以外にタイトでない複数個のレベルがある場合でも，何回かの交換を行ってもパッキングの高さをこれ以上低くすることができないことを示せば最適である．これを示すためには，一番上のレベル以外にタイトでないレベルがあるすべての場合を 1 つずつ検討する必要がある．

また，このアルゴリズムでは， p_{11} は最後に空き地も考慮して，BL で詰めることにしている．なぜならば， p_{11} は最小単位のピースであり，このピースより小さい空き地はなく，どんな形，どんな大きさの空き地にも p_{11} は必ずピッタリと詰めることができるからである．よって，最適性を考える場合， p_{11} はないものとして考えることにする．

(1) $W = 4$ (手続き W4) の場合

この場合は自明である (図 2. 2) ．

(2) $W = 5$ (手続き W5) の場合

この場合，step 2 で， p_{21} が p_{31} の最高レベルまで詰めることが

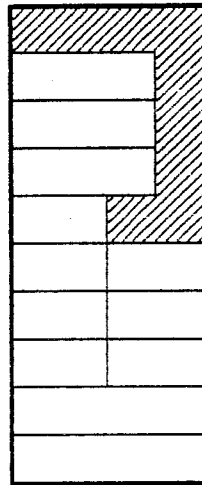


図 2. 2 アルゴリズム 2. 1 によるパッキング例 1
(手続き W 4)

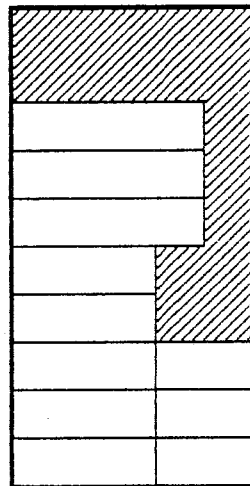


図 2. 3 アルゴリズム 2. 1 によるパッキング例 2
(手続き W 5)

できない場合でも，幅 2 以下のピースはないので最適である（図 2. 3）．

（3） $W \geq 6$ で， $W \bmod 4 = 0$ （手続き $Wm0$ ）の場合

この場合は W が 4 の倍数であるので，step1，step2 で， p_{41} ， p_{21} を BL で詰めたとき，一番上のレベルを除いてタイトである．

この状態で手続き $Wm0$ を終了し，次に手続き $exchange$ を行う． p_{31} を p_{21} ， p_{41} と交換する方法は， p_{31} 2 個と p_{21} 3 個， p_{31} 2 個と p_{21} ， p_{41} ， p_{31} 4 個と p_{41} 3 個の 3 通りのみが考えられる（図 2. 4）．手続き $exchange$ では，できる限り，かつタイトを崩さずにこの 3 通りの交換を行い，また p_{41} が p_{21} 2 個と交換できる場合はその交換を行った後 BL で詰める（図 2. 5）．ただし，step3 で p_{41} を p_{21} と交換しないのはできないからである．なぜな

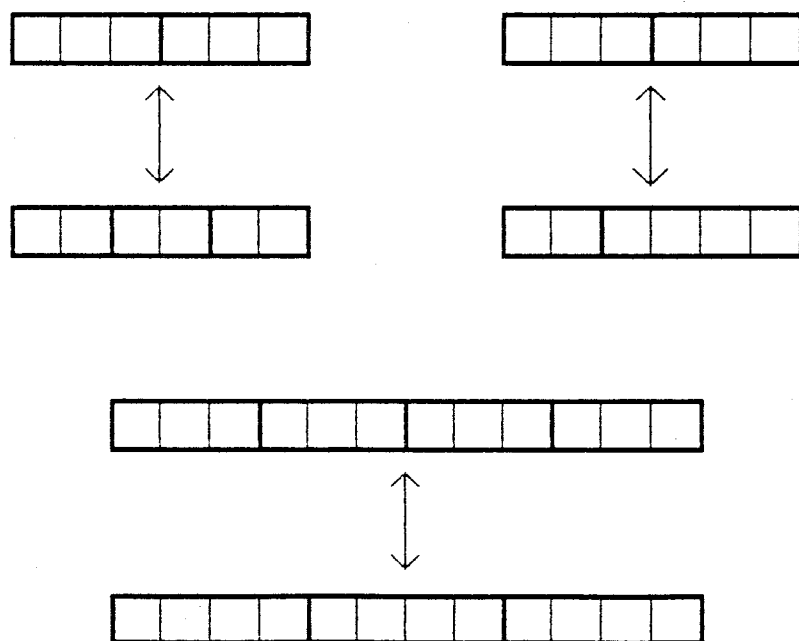


図 2. 4 p_{31} と p_{21} ， p_{41} との交換

らば、もし交換できるならばstep2を行っているはずである．よって、手続きexchange終了後は、一番上のレベルを除いてタイトであり、 p_{31} ができる限り p_{21} 、 p_{41} と交換されている．

ここで、 p_{21} 、 p_{41} が十分あり、 p_{31} が手続きexchangeでできる限り交換されるならば、 p_{31} は3個以下である．3個以下の場合は、手続きrepackで一番上のタイトでない（唯一の）レベルの続きに p_{31} をBLで詰めれば最適となる．なぜならば、 p_{31} が1個以下の場合は、明らかに最適に詰めることができる．また、 p_{31} が2個あるいは3個ある場合は、 p_{31} 2個と p_{41} 、幅2の空き地とを交換することができるが、 $W \geq 9$ のときは、 p_{31} 3個をBLで詰めてもた

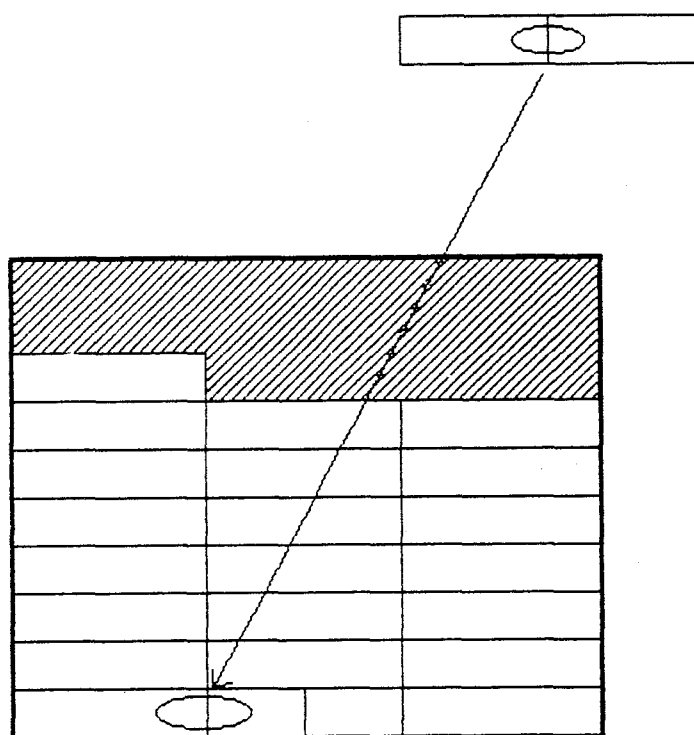


図 2. 5 アルゴリズム 2. 1 によるパッキング例 3
(手続きexchange)

かだかタイトでないレベルまでしか詰められず，この交換を行わなくても最適となる．また， $W=8$ の場合，幅2の空き地ができ，同じレベルに p_{41} がある場合は起こらない．なぜならば，この場合には必ず同じレベルに p_{21} があり，この p_{21} と p_{41} は既に手続き exchange で p_{31} 2 個と交換されているはずである（図2.6）．

次に，手続き repack で p_{31} が4個以上ある場合を考える．この場合が起こるのはもはやこれ以上 p_{31} を p_{21} ， p_{41} と交換できない場合であるから，タイトでないレベルには p_{41} がなくて p_{21} が2個以下，あるいは p_{21} がなくて p_{41} が2個以下の場合のみである．ここで問題となるのは p_{21} がなくて p_{41} が2個の場合である．この場合には p_{41} を2個先に詰めてしまうと，右端に幅2の空き地ができ反例となることがある（図2.7）．そこで，このように右端に幅2の空き地ができた場合は， p_{41} を1個取り除き幅6の空き地を作り，

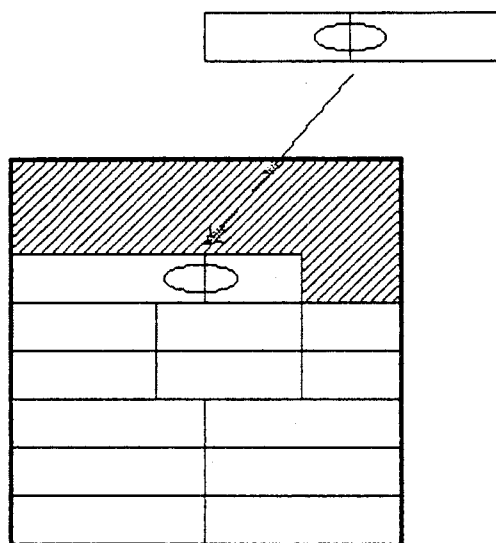


図2.6 アルゴリズム2.1によるパッキング例4
($W=8$ の場合)

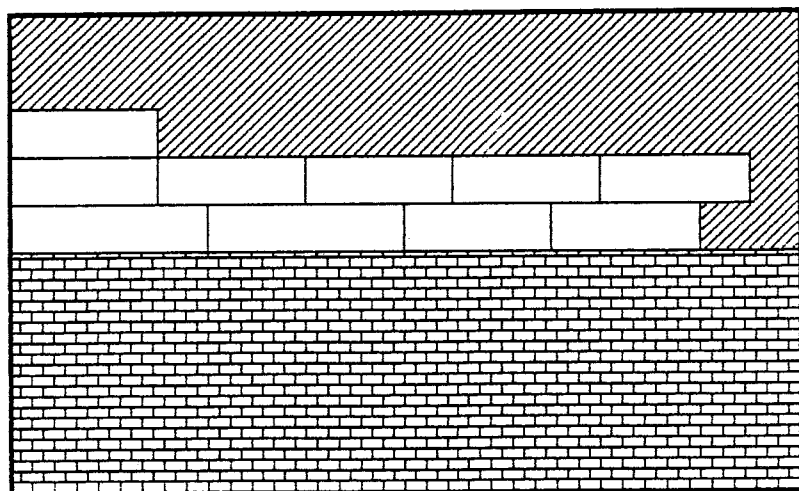


図 2. 7 アルゴリズム 2. 1 によるパッキング例 5
(手続き repack)

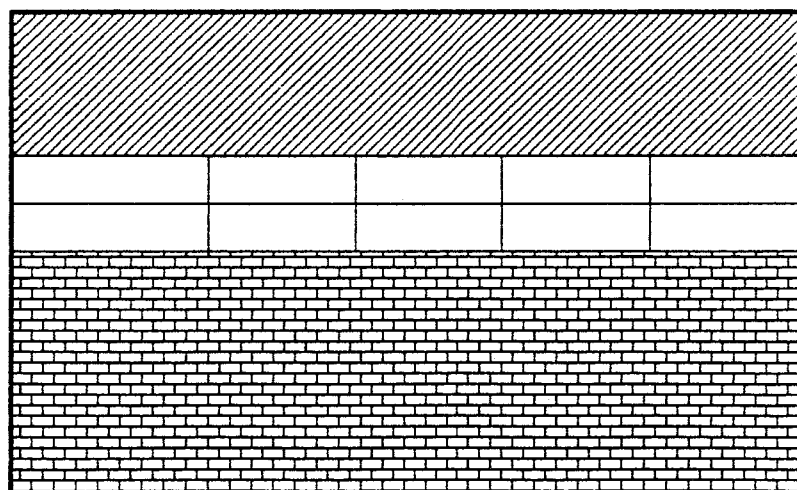


図 2. 8 アルゴリズム 2. 1 によるパッキング例 6
(手続き repack)

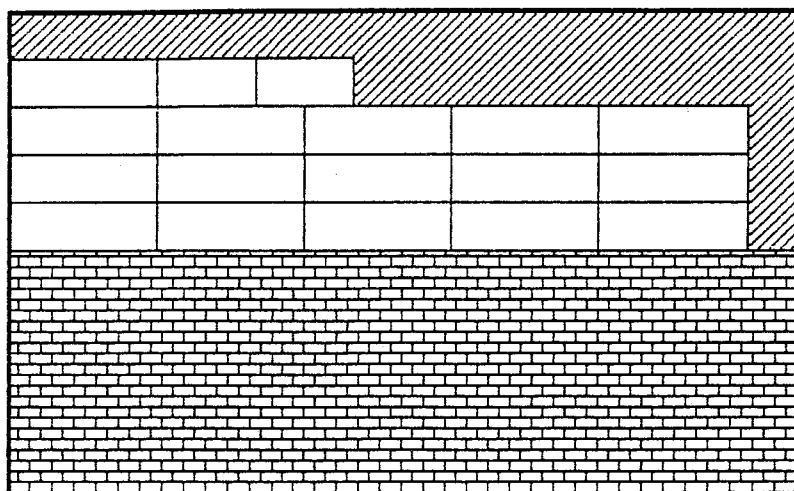


図 2. 9 アルゴリズム 2. 1 によるパッキング例 7
(手続き repack)

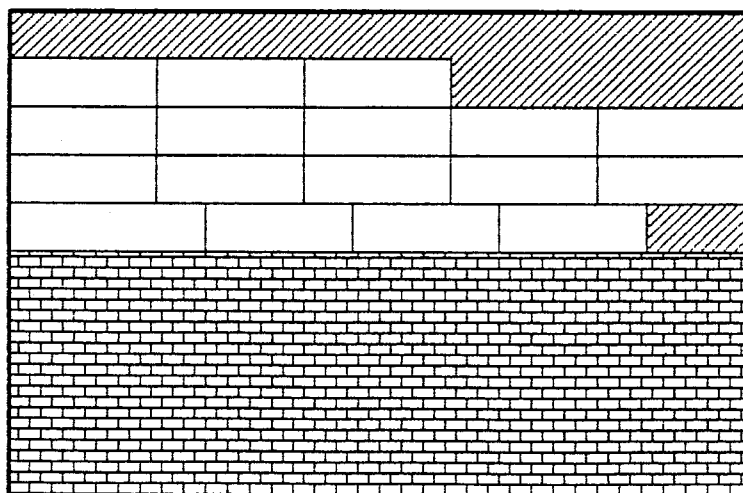


図 2. 10 アルゴリズム 2. 1 によるパッキング例 8
(手続き repack)

そこに p_{31} を 2 個詰めればそのレベルはタイトとなる．この上に p_{41} を BL で詰め，残りの p_{31} を BL で詰めると明らかに最適となる（図 2. 8）．次に， p_{41} がなくて p_{21} が 2 個以下の場合は， p_{31} ， p_{21} の順に BL で詰めれば明らかに最適となる（図 2. 9）．また， p_{21} がなくて p_{41} が 1 個以下の場合も， p_{41} ， p_{31} ， p_{21} の順に BL で詰めれば，空き地ができることもあるが最適となる（図 2. 10）．

よって， $W \geq 6$ で， $W \bmod 4 = 0$ の場合は最適に詰めることができる（図 2. 11）．

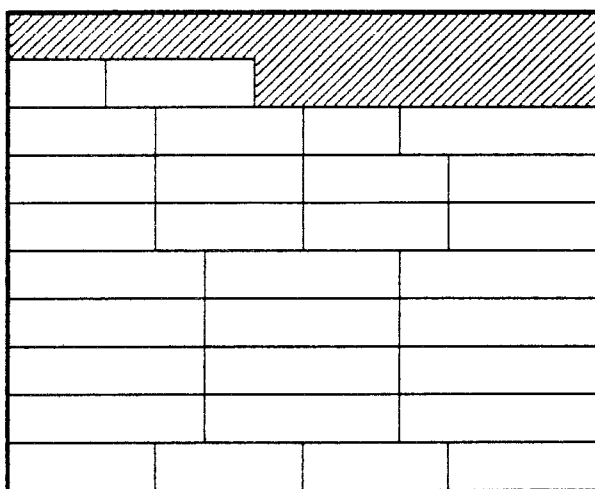


図 2. 11 アルゴリズム 2. 1 によるパッキング例 9
（手続き $Wm0$ ）

（4） $W \geq 6$ で， $W \bmod 4 = 1$ （手続き $Wm1$ ）の場合

この場合は W が奇数であるので，タイトに詰めるためには各レベルに p_{31} が必要である．そのためには，step 1 で p_{31} を右端に 1 列に詰める． p_{31} の左には幅 W' （ $W' \bmod 4 = 2$ ）のビンがあるこ

となる．step2で p_{41} を詰め，step3でその右にできた幅2の空き地に p_{21} を詰める．もし， p_{31} ， p_{21} がタイトな p_{41} の最高レベルまで詰めることができるならば，手続きWm1を行った後は，幅 W' の一番上のレベルを除いてタイトである．

ここで，もし p_{21} がタイトな p_{41} の最高レベルまで詰めることができなければ（step4），一番右の p_{41} 1個を取り除き，幅6の空き地を作り，そこに p_{31} を2個詰めることによってタイトとなるようにする（図2．12）．この操作をタイトな p_{41} の最高レベルまで，あるいは，右端に詰まっている p_{31} を上から順に2個取ると，その時点でタイトに詰まっている最高レベルの高さ（ H_t ）以下になるまで行う．この操作を行うごとに，その時点でタイトに詰まっている最高レベルの高さ（ H_t ）が1つ上がる．もし p_{31} が2個取

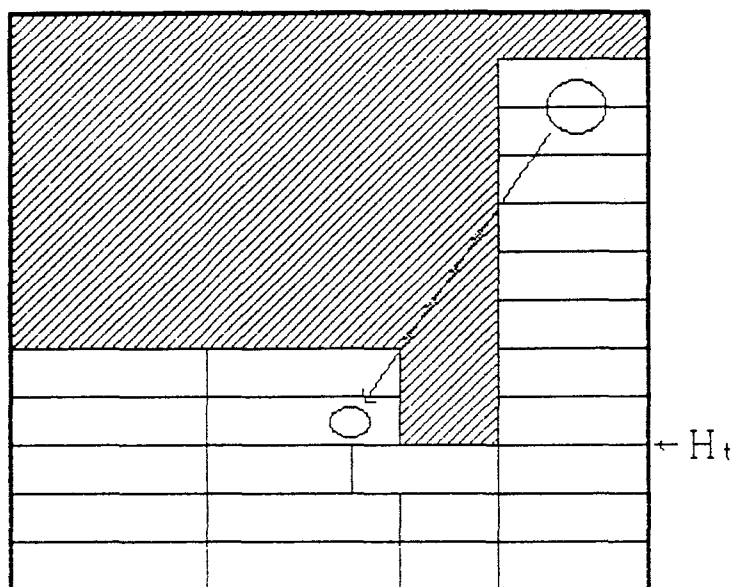


図2．12 アルゴリズム2．1によるパッキング例10
（手続きWm1）

るとタイトに詰まっている最高レベルの高さ以下になるならば、タイトに詰めることはできないので、上の複数個のレベルがタイトでないまま手続き W_{m1} を終了する。ここで、タイトに詰まっていないレベルの p_{31} はたかだか 2 個である。なぜならば、 p_{21} はないので、 p_{41} と p_{31} のみで 1 つのレベルをタイトにするためには、 $W \bmod 4 = 1$ の場合、 p_{31} が少なくとも 3 個必要だからである。次に、手続き $exchange$ を行う。この場合には、たかだか p_{31} は 2 個しかないので、交換はたかだか 1 回しか起こらない。次に、手続き $repack$ を行う。 p_{31} は 2 個以下であるので、step 1 で p_{41} 、 p_{31} 、 p_{21} の順に BL で詰める。この場合は、右端に空き地ができるが、 p_{41} と p_{31} 2 個以下とではタイトに詰めることができないので最適である（図 2. 13）。もし p_{31} が十分あり、 p_{41} を 1 個取り除くことによってできた幅 6 の空き地に p_{31} をタイトな p_{41} の最高レベルまで詰めることができるならば、手続き W_{m1} を行った後は、幅 W' の一番上のレベルを除いてタイトである。

次に、もし p_{31} がタイトな p_{41} の最高レベルまで詰めることができず、 p_{21} が p_{31} の最高レベルまで詰めることができるならば（step 5）、 p_{31} の最高レベルまで p_{21} を詰める。step 6 で残った p_{21} を BL で詰めて手続き W_{m1} を終了する。ただし、右端に空き地が残る（図 2. 14）ので手続き $repack$ で詰め直す。 p_{31} はないので手続き $exchange$ は行われぬ。次に、手続き $repack$ で上のタイトでないレベルに詰まっている p_{41} 、 p_{21} がその順で BL で詰められる。これは右端に幅 1 の空き地ができるが、 p_{31} がないので最適なパッキングである（図 2. 15）。

最後に、step 6 で残った p_{21} を BL で詰めるときに、 p_{31} が少なく p_{31} の上に p_{21} が詰められる場合がある。この場合も右端に幅 1 の空き地ができるが、 p_{31} がないので手続き $repack$ で最適に詰めることができる（図 2. 16）。

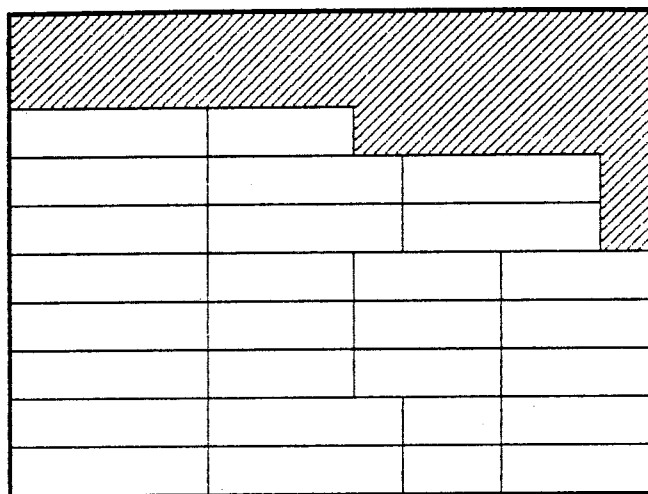


図 2. 1 3 アルゴリズム 2. 1 によるパッキング例 1 1
(手続き $W_m 1$)

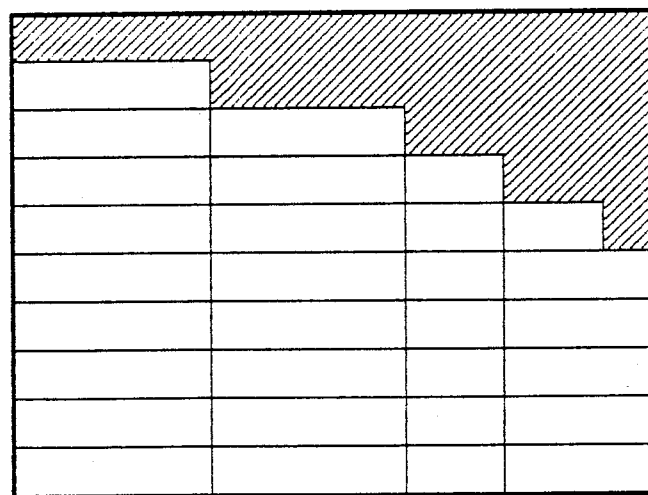


図 2. 1 4 アルゴリズム 2. 1 によるパッキング例 1 2
(手続き $W_m 1$)

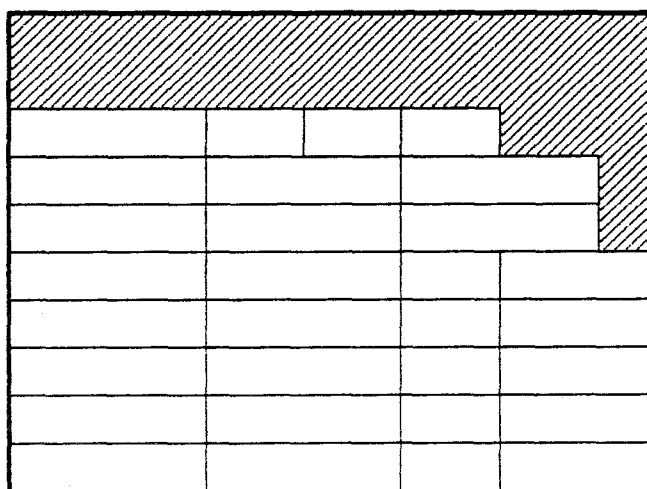


図 2. 1 5 アルゴリズム 2. 1 によるパッキング例 1 3
(手続き $W_m 1$)

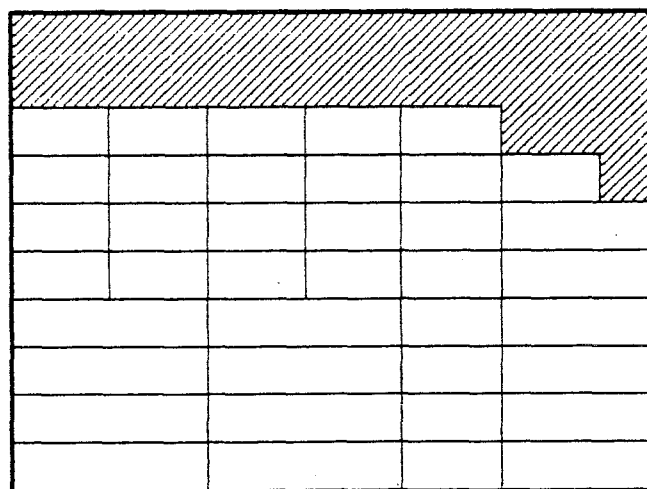


図 2. 1 6 アルゴリズム 2. 1 によるパッキング例 1 4
(手続き $W_m 1$)

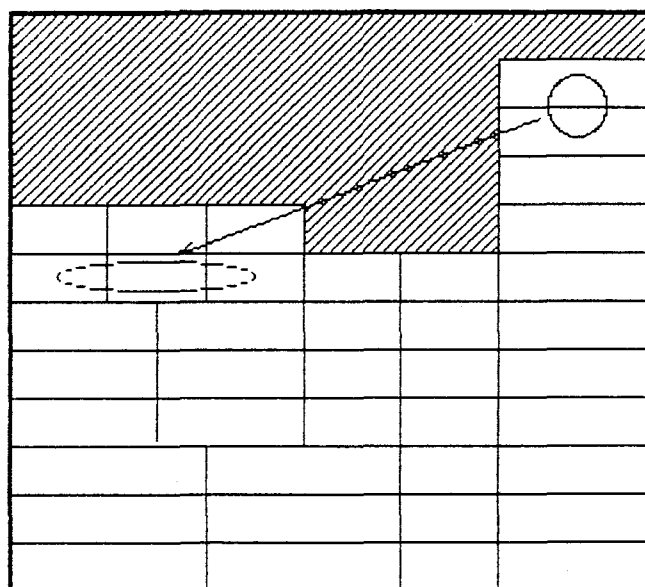


図2. 17 アルゴリズム2. 1によるパッキング例15
(手続きexchange)

幅 W' の一番上のレベルを除いてタイトである状態で手続き $Wm1$ を終了した場合は、手続きexchangeを行う（図2. 17）。この場合も（3）の場合と同様に、手続きexchange終了後は、幅 W' の一番上のレベルを除いてタイトであり、 p_{31} ができる限り p_{21} 、 p_{41} と交換されている。

ここで、 p_{21} 、 p_{41} が十分あり、 p_{31} が手続きexchangeでできる限り交換されるならば、 p_{31} は3個以下である。3個以下の場合も（3）の場合と同様に、手続きrepackで一番上のタイトでない（唯一の）レベルの続きに p_{31} をBLで詰めれば最適となる。また、手続きrepackで p_{31} が4個以上ある場合も（3）の場合と同様に、step2に従って詰めれば最適となる。

よって、 $W \geq 6$ で、 $W \bmod 4 = 1$ の場合は最適に詰めることができる（図 2. 18）。

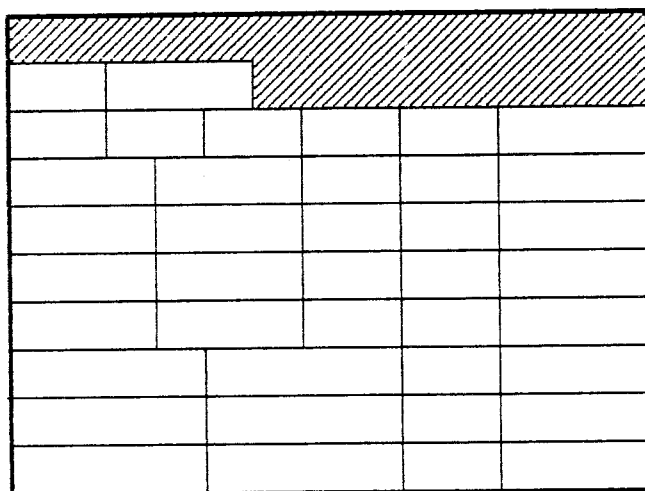


図 2. 18 アルゴリズム 2. 1 によるパッキング例 16
（手続き $Wm1$ ）

（5） $W \geq 6$ で、 $W \bmod 4 = 2$ （手続き $Wm2$ ）の場合

この場合、 W は偶数であるが 4 の倍数ではないので、step 1 で p_{41} を詰めたとき、右端に幅 2 の空き地ができる。step 2 でこの空き地に p_{21} を詰める。ここで、もし p_{21} がタイトな p_{41} の最高レベルまで詰めることができなければ（step 3）、一番右の p_{41} 1 個を取り除き、幅 6 の空き地を作り、そこに p_{31} を 2 個詰めることによってタイトとなるようにする（図 2. 19）。この操作をタイトな p_{41} の最高レベルまで、あるいは p_{31} が足りなくなる（1 個以下になる）まで行う。もし p_{31} も足りない（1 個以下）ならば、タイトに詰めることはできないので、上の複数個のレベルがタイトでないまま手続き $Wm2$ を終了する。次に、手続き exchange を行う。この

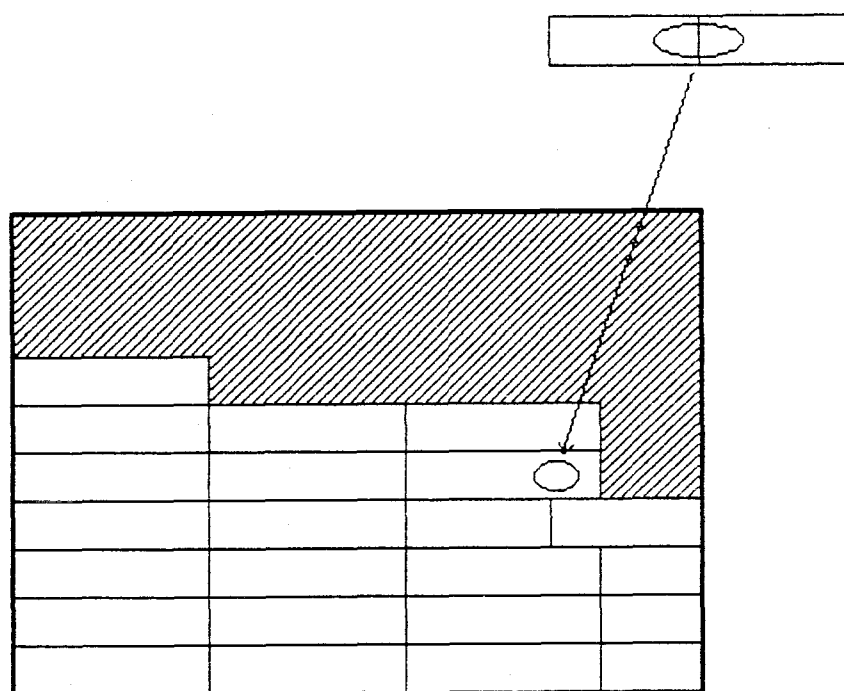


図 2 . 1 9 アルゴリズム 2 . 1 によるパッキング例 1 7
(手続き $Wm2$)

場合には、 p_{31} はただか1個しかないので、交換は起こらない。次に、手続きrepackを行う。 p_{31} は1個以下であるので、step1で p_{41} 、 p_{31} の順にBLで詰める。この場合は、右端に空き地ができるが、 p_{41} と p_{31} 1個以下とではタイトに詰めることができないので最適である（図2.20）。もし p_{31} が十分あり、 p_{41} を1個取り除くことによってできた幅6の空き地に p_{31} をタイトな p_{41} の最高レベルまで詰めることができるならば、手続きWm2を行った後は、一番上のレベルを除いてタイトである。

また、 p_{21} が十分あり、タイトな p_{41} の最高レベルまで詰めることができるならば、step4で残った p_{21} をBLで詰める。この場合も手続きWm2を行った後は、一番上のレベルを除いてタイトである。

一番上のレベルを除いてタイトである状態で手続きWm2を終了した場合は、手続きexchangeを行う（図2.21）。この場合も

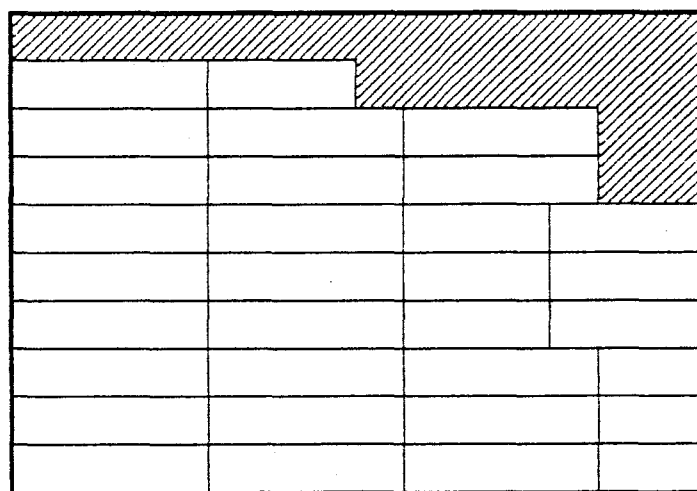


図2.20 アルゴリズム2.1によるパッキング例18
(手続きWm2)

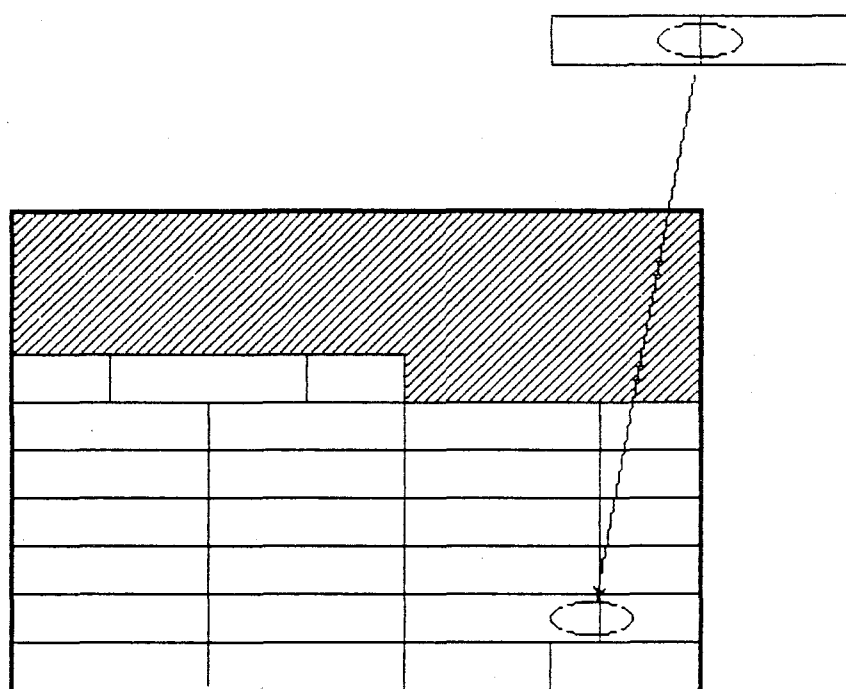


図 2. 2 1 アルゴリズム 2. 1 によるパッキング例 1 9
(手続き exchange)

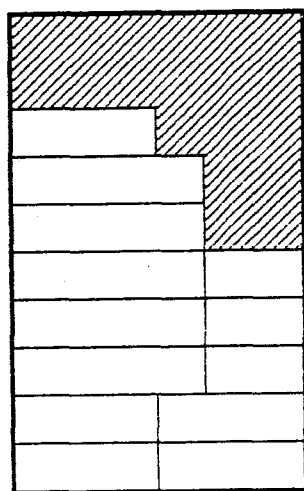


図 2. 2 2 アルゴリズム 2. 1 によるパッキング例 2 0
($W = 6$ の場合)

(3) の場合と同様に，手続きexchange終了後は，一番上のレベルを除いてタイトであり， p_{31} ができる限り p_{21} ， p_{41} と交換されている．

ここで， p_{21} ， p_{41} が十分あり， p_{31} が手続きexchangeでできる限り交換されるならば， p_{31} は3個以下である．3個以下の場合も(3)の場合と同様に， $W \geq 9$ のときは，手続きrepackで一番上のタイトでない(唯一の)レベルの続きに p_{31} をBLで詰めれば最適となる．また， $W = 6$ の場合も， W が3の倍数で p_{31} がタイトに詰まるので， p_{31} を後で詰めても最適となる(図 2. 2 2)．また，手続きrepackで p_{31} が4個以上ある場合も(3)の場合と同様に，step 2に従って詰めれば最適となる．

よって， $W \geq 6$ で， $W \bmod 4 = 2$ の場合は最適に詰めることができる(図 2. 2 3)．

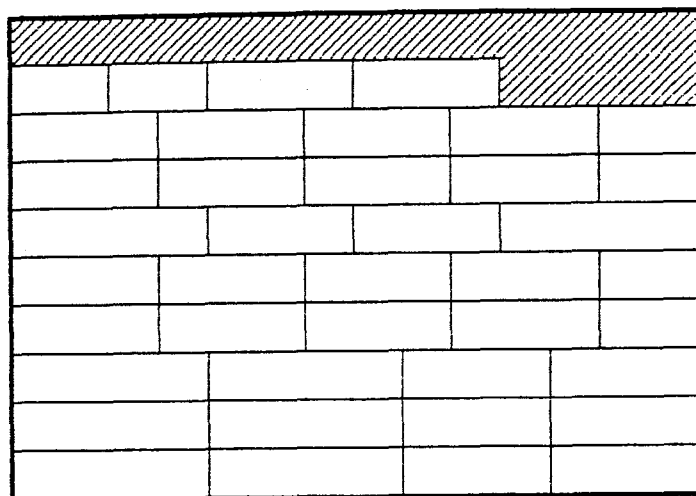


図 2. 23 アルゴリズム 2. 1 によるパッキング例 2 1
(手続き $Wm2$)

(6) $W \geq 6$ で, $W \bmod 4 = 3$ (手続き $Wm3$) の場合

この場合は W が奇数であるので, タイトに詰めるためには各レベルに p_{31} が必要である. そのためには, step1 で p_{31} を右端に 1 列に詰める. p_{31} の左には幅 W' ($W' \bmod 4 = 0$) のビンがあることになる. 次に, step2, step3 で, p_{41} , p_{21} を BL で詰める. もし, p_{31} が十分あり, タイトな p_{41} , p_{21} の最高レベルまで詰めることができるならば, 手続き $Wm3$ を行った後は, 幅 W' の一番上のレベルを除いてタイトである.

次に, もし p_{31} がタイトな p_{41} の最高レベルまで詰めることができないならば, step3 で右端の幅 3 の空き地に p_{21} を BL で詰める. この場合, 右端に幅 1 の空き地ができるが, p_{31} がないので最適である. また, もし p_{21} もタイトな p_{41} の最高レベルまで詰めることができない場合も, 右端に幅 3 の空き地ができるが p_{41} しかないの

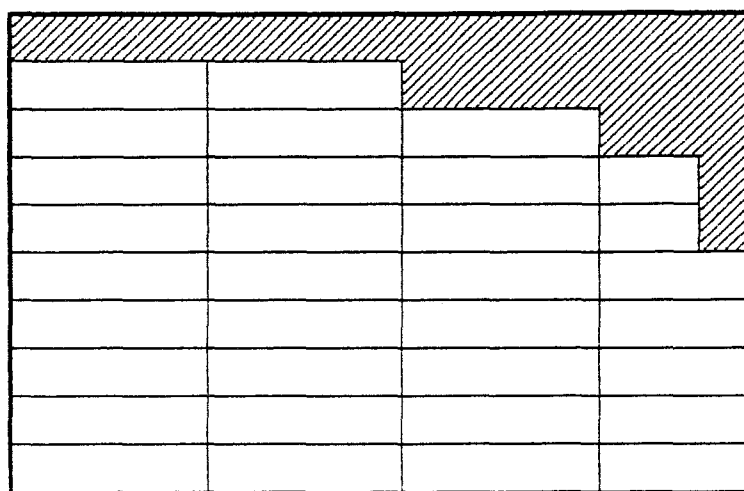


図 2. 2 4 アルゴリズム 2. 1 によるパッキング例 2 2
(手続き Wm 3)

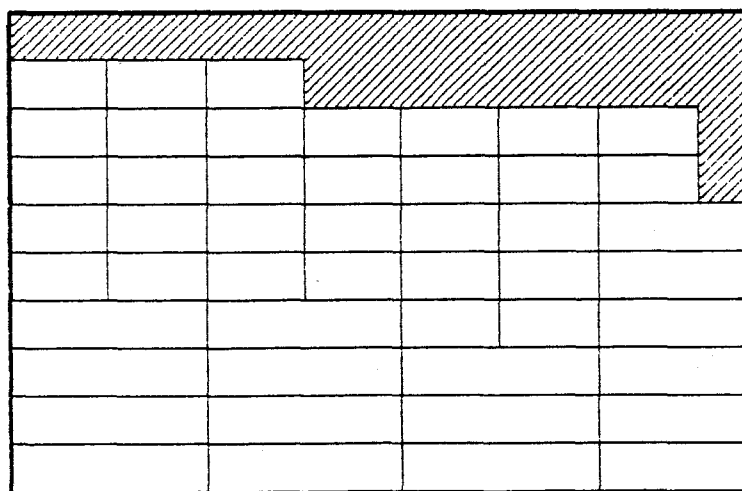


図 2. 2 5 アルゴリズム 2. 1 によるパッキング例 2 3
(手続き Wm 3)

で最適である（図 2. 24）．また， p_{31} がタイトな p_{41} の最高レベルまで詰めることができたが，step3 を実行中に p_{21} が p_{31} のレベルを越えてしまった場合でも，右端に幅 1 の空き地ができるが p_{31} がないので最適である（図 2. 25）．

幅 W' の一番上のレベルを除いてタイトである状態で手続き $Wm3$ を終了した場合は，手続き exchange を行う（図 2. 26）．この場合も（3）の場合と同様に，手続き exchange 終了後は，幅 W' の一番上のレベルを除いてタイトであり， p_{31} ができる限り p_{21} ， p_{41} と交換されている．

ここで， p_{21} ， p_{41} が十分あり， p_{31} が手続き exchange でできる限り交換されるならば， p_{31} は 3 個以下である．3 個以下の場合も（3）の場合と同様に， $W \geq 9$ のときは，手続き repack で一番上の

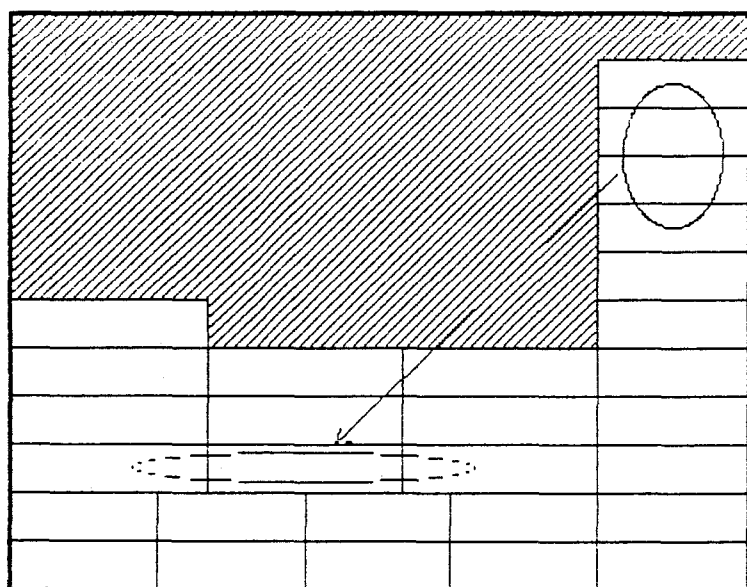


図 2. 26 アルゴリズム 2. 1 によるパッキング例 24
（手続き exchange）

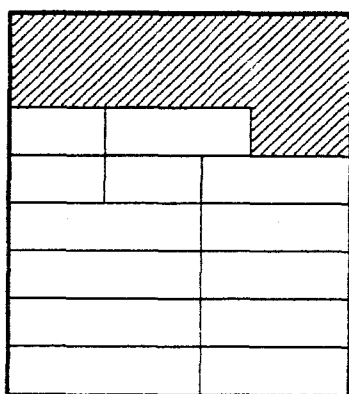


図 2. 2 7 アルゴリズム 2. 1 によるパッキング例 2 5
($W = 7$ の場合)

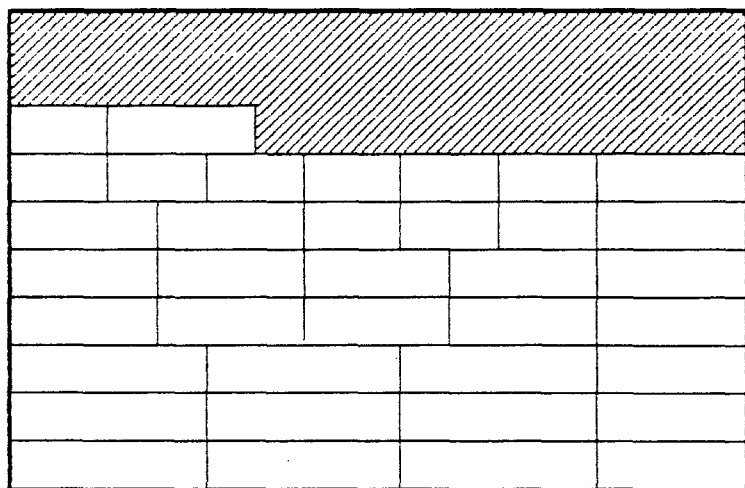


図 2. 2 8 アルゴリズム 2. 1 によるパッキング例 2 6
(手続き $Wm3$)

タイトでない（唯一の）レベルの続きに p_{31} を BL で詰めれば最適となる．また， $W = 7$ の場合も，幅 2 の空き地ができ同じレベルに p_{41} がある場合は起こらないので，最適となる（図 2. 27）．また，手続き repack で p_{31} が 4 個以上ある場合も（3）の場合と同様に，step 2 に従って詰めれば最適となる．

よって， $W \geq 6$ で， $W \bmod 4 = 3$ の場合は最適に詰めることができる（図 2. 28）．

（7） アルゴリズムの時間計算量

このアルゴリズムが $O(n)$ 回の BL パッキングで行うことができることを示す．まず， $W = 4$ ， $W = 5$ の場合は詰め直すことを行わないので， n 回の BL パッキングで行うことができる． $W \geq 6$ の場合は，まず，手続き $Wm1$ ，手続き $Wm2$ で， p_{41} を詰め直すことがあるが， p_{31} 2 個と p_{41} を交換するので，この回数は $1/2 \cdot n_{31}$ 回以下である．また，手続き exchange で p_{21} ， p_{41} を詰め直すことがあるが，これはたかだか p_{41} と p_{21} 2 個との交換， p_{41} ， p_{21} と p_{31} 2 個との交換，そして p_{41} と p_{21} 2 個との交換だけであるから，この回数は $7/2 \cdot n_{31}$ 回以下である．また，手続き repack ではすべてのピースをもう 1 度だけ詰め直すことがある．なお，ピースを交換する際，交換すべきピースを見つける操作は各幅のピースが何個，どのレベルにあるかを覚えておくことにより，定数時間でできる．以上のことから，BL パッキングはたかだか $(2 \cdot n + 4 \cdot n_{31})$ 回以下である．よって，このアルゴリズムは $O(n)$ 回の BL パッキングで行うことができる． $\square$

定理 2. 1 から，ピースの大きさを 4 以下に限定した場合の 1 次元ビン・パッキング問題は，多項式時間で解けることになる．

2. 3 ピースの幅が一定である場合

ピースの幅が一定であるという制約を加えると，問題はスケジュー

ーリング問題における最大滞留時間最小問題^[4]となる。この問題に関しては次の定理が導ける。

〔定理 2. 2〕 ピースの種類を $h \times 1$ の部分矩形に限定した場合の厳密解法が存在するとする。このとき、この解法を利用して、ピースの種類を $1 \times h$ の部分矩形に限定した場合の最適解を求めるアルゴリズム（アルゴリズム 2. 2）が存在する。ただし、 $h \times 1$ の部分矩形に対する厳密解法の時間計算量が $O(f(n))$ であるならば、この厳密解法の時間計算量は $O(h \cdot f(n))$ となる。なお、 n はピース数である。 □

アルゴリズム 2. 2

- step 0 このアルゴリズムにおいて $h \times 1$ の部分矩形に対する厳密解法を行う場合は、 $1 \times h$ の部分矩形であるすべてのピースを、幅と高さを交換して $h \times 1$ の部分矩形であるとする；
- step 1 ピースの面積の総和をビンの幅 W で割ると下界が求まる； それを L とする；
- step 2 repeat
- ・ L をビンの幅と考えて、 $h \times 1$ の部分矩形に対する厳密解法を行う；
 - ・ もし最適解 H_{opt} が実際のビンの幅 W を越えるならば、 L を 1 つ増やす；
- until 最適解 H_{opt} が実際のビンの幅 W 以下となる；

（証明） このアルゴリズムで求まる解は、 $h \times 1$ の部分矩形に対する厳密解法を使って、下界から L （実際のビンの高さ）を 1 つずつ増やしていくので、明らかに最適である（図 2. 29）。

また、 $1 \times h$ の部分矩形の最適なパッキングの中には、最適解

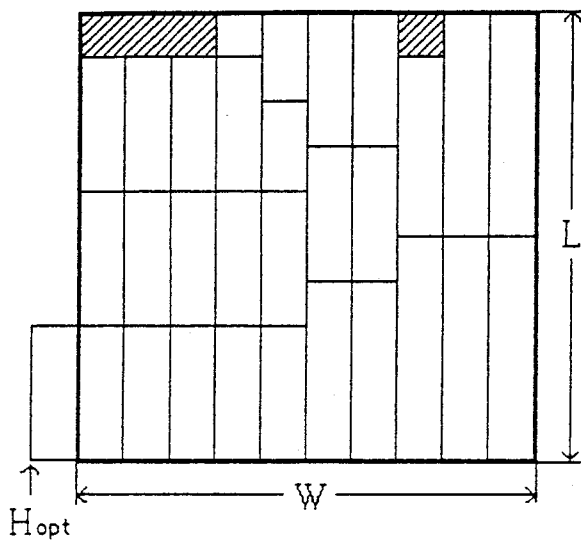


図 2. 29 アルゴリズム 2. 2 によるパッキング例

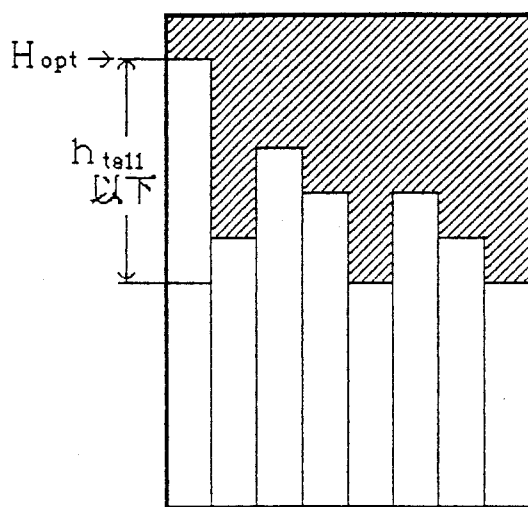


図 2. 30 $1 \times h$ の部分矩形の最適なパッキング例

H_{opt} よりたかだか h_{tail} だけ低い部分を除いてタイトに詰まっているものが、少なくとも 1 つある (図 2.30)。ただし、 h_{tail} は最も高いピースの高さである。したがって、最適解は、下界 + h_{tail} 未満である。ゆえに、 $1 \times h$ の部分矩形に限定した場合、step 2 の操作はたかだか h 回しか行われぬ。よって、このアルゴリズムの時間計算量は $O(h \cdot f(n))$ となる。□

ピースの種類を 4×1 の部分矩形に限定した場合の厳密解法 (アルゴリズム 2.1) が存在するので、この定理から次の系が導かれる。

〔系 2.1〕 ピースの種類を 4×1 の部分矩形に限定した場合の厳密解法 (アルゴリズム 2.1) を利用して、ピースの種類を 1×4 の部分矩形に限定した場合の最適解を求めるアルゴリズムが存在する。また、このアルゴリズムは $O(n)$ 回の BL パッキングで行うことができる。ただし、 n はピース数である。□

系から、ジョブの処理時間を 4 以下に限定した場合の最大滞留時間最小問題は、多項式時間で解けることになる。

2.4 ピースの種類が 2×2 の部分矩形である場合

本節では、ピースの種類を $w \times h$ の部分矩形に限定し、 w および h をパラメータとして多項式時間の厳密解法が考えられるかについて考察する。そして、ピースの種類を 2×2 の部分矩形に限定した場合の厳密解法を提案し、次の定理を導いている。

〔定理 2.3〕 ピースの種類を 2×2 の部分矩形に限定する。このとき、ピースをアルゴリズム 2.3 で詰めると、必ず最適に詰めることができる。また、このアルゴリズムは $O(n)$ 回の BL パッキングで行うことができる。ただし、 W はビンの幅であり、 n はピース数である。□

アルゴリズム 2. 3

- step1 p_{22} をBLで詰める； もし右端（Wが奇数のときは右端の1つ左）まで詰めることができないならば，そのレベルのピースはすべて詰めないでおく； すなわち， p_{22} を $\lfloor n_{22} / \lfloor W / 2 \rfloor \rfloor$ 列をBLで詰め，その高さは $2 \cdot \lfloor n_{22} / \lfloor W / 2 \rfloor \rfloor$ である；
- step2 p_{21} をBLで詰める； もし右端（Wが奇数のときは右端の1つ左）まで詰めることができないならば，そのレベルのピースはすべて詰めないでおく； また，もし詰めたレベルが奇数の場合は，一番上のレベルのピースはすべて詰めないで置き，高さを偶数にする； すなわち， p_{21} を $2 \cdot \lfloor n_{21} / (2 \cdot \lfloor W / 2 \rfloor) \rfloor$ 列詰め，その高さは $2 \cdot \lfloor n_{21} / (2 \cdot \lfloor W / 2 \rfloor) \rfloor$ である；
- step3 もしWが奇数ならば， p_{12} を右端の幅1の空き地に左のピースの高さまで詰める；
- step4 残った p_{22} をBLで詰める；
- step5 p_{12} をBLで詰める；
- step6 残った p_{21} をBLで詰める；
- step7 p_{11} をBLで詰める；

（証明） このアルゴリズムでは，[定理 2. 1] の証明の場合と同様に， p_{11} はないものとして考える．

step1，step2では，Wが偶数のときはタイトなパッキングである．Wが奇数のとき，step3で p_{12} を右端の幅1の空き地に詰める．左に詰められているピースの高さは偶数であるので，もし p_{12} が十分あり，左のピースの高さまで詰めることができるならば，step3終了後はWが奇数の場合もタイトに詰められる．もし p_{12} が左のピースの高さまで詰めることができないならば，右端に幅1の空き地

ができるが，幅 1 のピースがないので最適なパッキングである（図 2.31）．

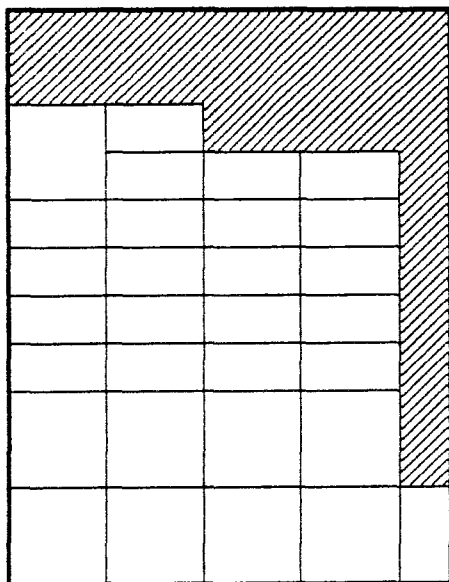


図 2.31 アルゴリズム 2.3 によるパッキング例 1

step 4 では，1 列に満たない（たかだか $\lfloor W/2 \rfloor - 1$ 個の） p_{22} を BL で詰め，step 5 では，その続きに p_{12} を BL で詰める．ここまでは明らかにタイトなパッキングである．次に，（たかだか $2 \cdot \lfloor W/2 \rfloor - 1$ 個の） p_{21} を BL で詰めるのであるが，このとき W が偶数，奇数であるに関わらず，右端に 1×2 の空き地ができる可能性がある．しかし，残った p_{21} を詰めたときの高さは，たかだか 2 であるので（ $\because p_{21}$ はたかだか $2 \cdot \lfloor W/2 \rfloor - 1$ 個）， p_{12} を 1 つ取り除き，そこに p_{21} を 2 つ詰め，空き地をなくしてしまい， p_{12} をその上のレベルに詰めるという操作をとる必要はない．ゆえに，この場合も最適なパッキングである（図 2.32）．

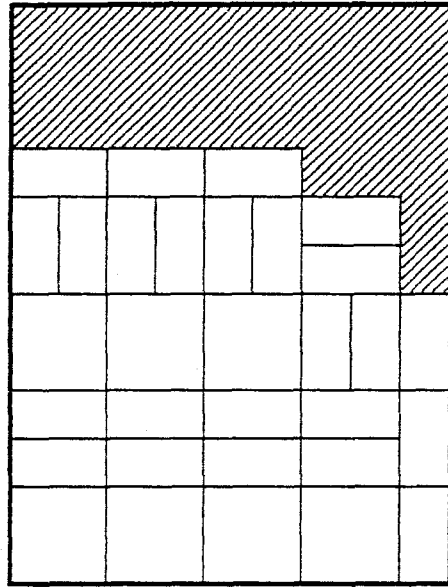


図 2. 3 2 アルゴリズム 2. 3 によるパッキング例 2

このアルゴリズムでは，step 1 で残った，たかだか $\lfloor W/2 \rfloor - 1$ 個のピースの詰め直しと，step 2 で残った，たかだか $2 \cdot \lfloor W/2 \rfloor - 1$ 個のピースの詰め直しがある（これは，前もって，ピースの数からレベルの数を計算することによって，避けることができる）だけで，それ以外に詰め直しは起こらない．したがって，このアルゴリズムは $O(n)$ 回の BL パッキングで行うことができる． □

2. 5 結言

本章では，ビンの幅，ピースの幅，高さを整数値に制約した 2 次元ビン・パッキング問題を取り上げ，その厳密解法を考察した．この制約のもとで，ピースの種類を $w \times h$ の部分矩形に限定すること

によって，線形時間で最適解が得られるかどうかについて考察した．その結果，少なくともピースの種類が 4×1 ， 1×4 ， 2×2 の部分矩形である場合までは，線形時間で最適解が得られることを明らかにした．

第3章 ピースに制約を加えた 2次元ビン・パッキング 問題の近似解法

3.1 緒言

本章では，ピースに制約を加えた2次元ビン・パッキング問題に対する3つの近似解法を提案し，その良さを評価する． $w \times h$ の部分矩形において w および h の値が前章で見い出した厳密解法の整数値を越えると，もはや時間計算量がピース数 n の線形時間となる厳密解法は期待できそうにない状況にあり，近似解法を主題とせざるをえないと思われる．このような場合でも，前章で見い出した厳密解法の考え方や解法そのものを活用することにより，良い近似解法を構成することができる可能性があると思われる．

まず，近似解法の最悪の場合の理論的解析を考え，ピースの種類を 3×2 の部分矩形に限定した場合の近似解法を提案する．この近似解法は前章のピースの種類が 4×1 の部分矩形である場合の厳密解法の考え方を基にしたものであり，線形時間で最悪の場合でもたかだか高さが1しか悪くならない近似解が得られることを明らかにする．

次に，ピースの種類を $w \times 1$ および $w \times 2$ ($w \geq 5$)の部分矩形に限定した場合の近似解法を提案し，その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価する．ピースの種類が $w \times 1$ の部分矩形である場合の近似解法は，前章のピースの種類が 4×1 の部分矩形である場合の厳密解法を活用したものであり，ピースの種類が $w \times 2$ の部分矩形である場合の近似解法は，前述のピースの種類が 3×2 の部分矩形である場合の近似解法の考え方を基に，ピースの種類が $w \times 1$ の部分矩形である場合の近似解法を活用したものである．

3. 2 近似解法の理論的解析

～ピースの種類が 3×2 の部分矩形である場合

本節では、ピースの種類を 3×2 の部分矩形に限定した場合について議論する。この場合には常に最適解が得られるとは限らないが、最悪の場合でもたかだか高さが1しか悪くならない近似解法を提案し、次の定理を導いている。

【定理3. 1】 ピースの種類を 3×2 の部分矩形に限定する。このとき、ピースをアルゴリズム3. 1で詰めると、近似度は次のようになる。

$$H_A \leq H_{opt} + 1$$

また、このアルゴリズムは $O(n)$ 回のBLパッキングで行うことができる。ここで、 H_A は近似解法でのパッキングの高さ、 H_{opt} は最適なパッキングの高さである。ただし、 W はビンの幅であり、 n はピース数である。 □

アルゴリズム3. 1

メイン・ルーチン

- step1 幅2以上の高さ1のピース (p_{31} , p_{21}) を縦に2つ並べて、高さ2のピースとする； ただし、 n_{31} , n_{21} が奇数となり、ピースが1つ余る場合は、余ったピースはstep7で詰めるので、そのまま残しておく；
- step2 if $W = \text{偶数}$ then 手続きevenを行う；
- step3 if $W = \text{奇数}$ then 手続きoddを行う；
- step4 if (W が偶数のとき, $W \geq 6$) or
(W が奇数のとき, $W \geq 9$) then
手続きexchangeを行う；
- step5 タイトに詰まっていないレベルで、
手続きrepackを行う；

- step6 p_{12} をBLで詰める；
- step7 残ったピース（たかだか， p_{31} と p_{21} が1ずつ）をBLで詰める；
- step8 p_{11} をBLで詰める；

手続きeven

- step1 p_{22} をBLで詰める；

手続きodd

- step1 p_{32} を右端に1列に詰める；
- step2 p_{22} をBLで詰める；

手続きexchange

- step0 この手続きにおいて交換される p_{32} は， W が偶数のときはフリーの p_{32} であり， W が奇数のときは右端に1列に詰められた p_{32} である； このピースを p_{ex} とする；
 W が偶数のときは，交換することができるフリーの p_{32} がなくなる（2個未満になる）まで，以下の操作を行う； また， W が奇数のときは，交換するために右端の一番上の p_{32} を取り除くことによって，その時点でタイトに詰まっている最高レベルの高さ（ H_t ）未満になるならば，交換することを行わず，この手続きは終了する；
- step1 同じレベルの p_{22} 3個を p_{ex} 2個と交換し，交換された p_{22} 3個をBLで詰める；

手続きrepack

- step1 残った p_{32} が1個以下ある場合は， p_{22} ， p_{32} の順に

BLで詰める；
 step2 残った p_{32} が 2 個以上ある場合は， p_{32} ， p_{22} の順に
 BLで詰める；

(証明) このアルゴリズムでは，[定理 2. 1] の証明の場合と同様に， p_{11} はないものとして考える．

(1) $W = \text{偶数}$ (手続き even) の場合

この場合は W が偶数であるので，手続き even の step 1 で， p_{22} を

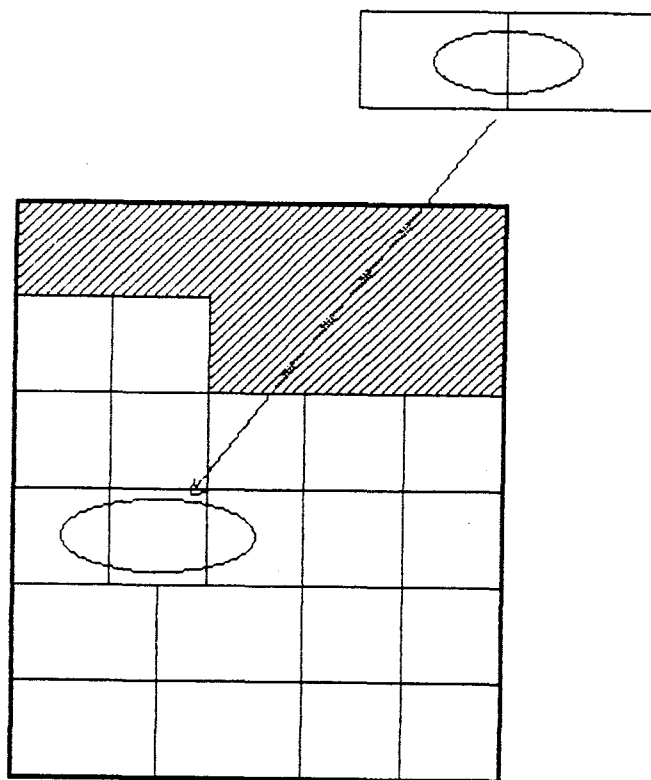


図 3. 1 アルゴリズム 3. 1 によるパッキング例 1
 (手続き exchange)

BLで詰めたとき，一番上のレベルを除いてタイトである．

次に， $W \geq 6$ のとき手続きexchangeを行い，できる限り，かつタイトを崩さずに， p_{32} を p_{22} と交換する（図3．1）． p_{32} を p_{22} と交換する方法は， p_{32} 2個と p_{22} 3個との交換のみが考えられる．よって，手続きexchange終了後も，一番上のレベルを除いてタイトであり， p_{32} ができる限り p_{22} と交換されている．

ここで， p_{22} が十分あり， p_{32} が手続きexchangeでできる限り交換されているならば，手続きrepackで残った p_{32} は1個以下である． p_{32} が1個以下ならば，手続きrepackで一番上のタイトでない（唯一の）レベルの続きに p_{32} をBLで詰める．この場合は，一般には，一番上の高さ2のレベルを除いてタイトとなるが，上の2つの高さ2のレベルがタイトとならない場合がある（図3．2）．しかし，

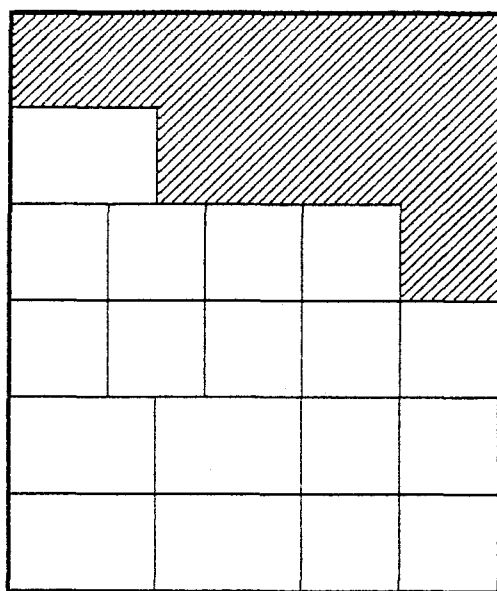


図3．2 アルゴリズム3．1によるパッキング例2
（手続きrepack）

この場合でも面積の下限から考えて、この解法によるパッキングの高さは最適解に比べてたかだか1しか悪くならないので近似度は保たれる。その後、step6で p_{12} がBLで詰められ、step7で残っている（たかだか1個の） p_{31} 、 p_{21} がBLで詰められる。この場合も、一般に、一番上の高さ2のレベルを除いてタイトである。ただし、step7で p_{31} 、 p_{21} を詰めるとき、高さ3のタイトでない領域を作る可能性がある（図3.3）。しかし、この場合も面積の下限から考えて、この解法によるパッキングの高さは最適解に比べてたかだか1しか悪くならないので近似度は保たれる。

次に、手続きrepackで残った p_{32} が2個以上ある場合を考える。この場合が起こるのは、もうこれ以上 p_{32} を p_{22} と交換できない場合であり、上のタイトでないレベルには p_{22} は2個以下である。こ

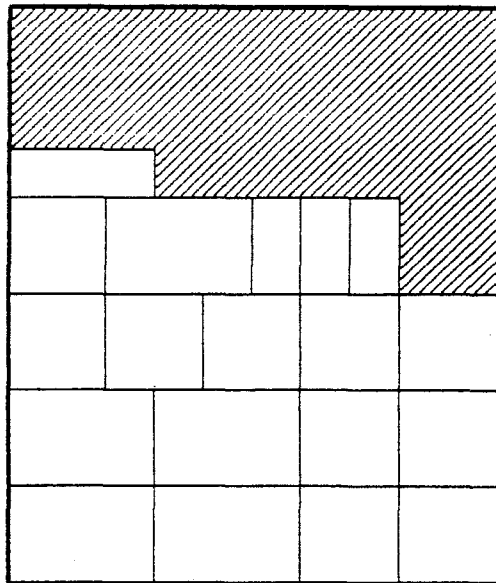


図3.3 アルゴリズム3.1によるパッキング例3
(手続きeven)

の場合は，たとえすべてのピースの高さが1であると考えても，たかだか1回を除いて幅3のピースと幅2のピースの交換はできないので，手続きrepackで p_{32} ， p_{22} を順にBLで詰めれば近似度は保たれる．また， p_{21} 2個と p_{31} ，幅1の空き地との交換がたかだか2回考えられるが，これを行ってもパッキングの高さは最適解に比べてたかだか1しか悪くならないので近似度は保たれる．その後，step6で p_{12} がBLで詰められ，step7で残っている（たかだか1個の） p_{31} ， p_{21} がBLで詰められる．この場合， p_{31} と p_{21} ，幅1の空き地を交換して空き地を減らすことができるが，それを行わなくてもパッキングの高さは最適解に比べてたかだか1しか悪くならないので近似度は保たれる（図3.4）．

以上のことから， W = 偶数の場合にこのアルゴリズムで詰めると，

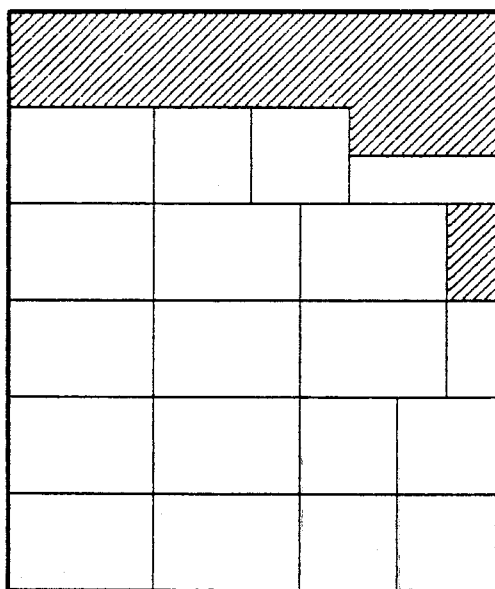


図3.4 アルゴリズム3.1によるパッキング例4
(手続きeven)

近似度は $H_A \leq H_{opt} + 1$ となる (図 3. 5) .

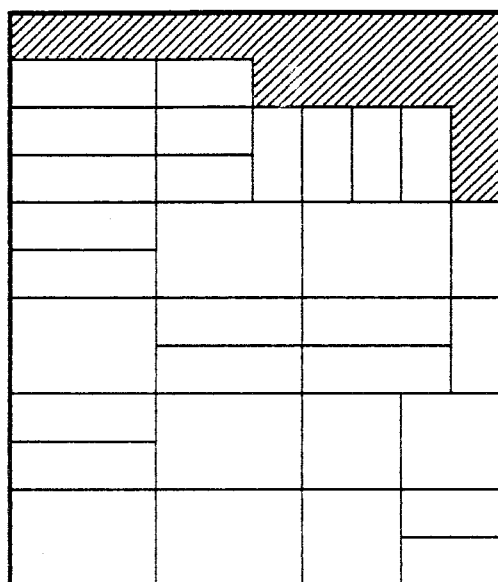


図 3. 5 アルゴリズム 3. 1 によるパッキング例 5
(手続き even)

(2) $W =$ 奇数 (手続き odd) の場合

この場合には W が奇数であるので、タイトに詰めるためには各レベルに p_{32} が必要である。そのために手続き odd の step 1 で p_{32} を右端に 1 列に詰める。 p_{32} の左には、幅 W' ($W' =$ 偶数) のビンがあることになる。手続き odd の step 2 では、 p_{22} を BL で詰める。ここで、 p_{32} が十分あり、タイトな p_{22} の最高レベルまで詰めることができるならば、手続き odd を行った後は、幅 W' の一番上のレベルを除いてタイトである。

次に、もし p_{32} がタイトな p_{22} の最高レベルまで詰めることができないならば、手続き odd の step 2 で p_{32} の上に p_{22} を詰めること

になり，右端に幅 1 の空き地ができる．しかし， p_{32} はないので p_{12} 以外で空き地をなくすることはできない．この場合，手続き exchange は行われずに，手続き repack へ進む．手続き repack でも， p_{22} しかないのので右端に幅 1 の空き地ができたまま詰められる．step 6 で p_{12} が BL でまず右端の幅 1 の空き地に詰められ，step 7 で残っている p_{31} ， p_{21} が BL で詰められる．この場合， p_{31} と p_{21} ，幅 1 の空き地を交換して空き地を減らすことができるが，それを行わなくてもパッキングの高さは最適解に比べてただか 1 し悪くならないので近似度は保たれる（図 3. 6）．

幅 W' の一番上のレベルを除いてタイトである状態で手続き odd を終了した場合は， $W \geq 9$ のとき手続き exchange を行う（図 3. 7）．この場合も（1）の場合と同様に，手続き exchange 終了

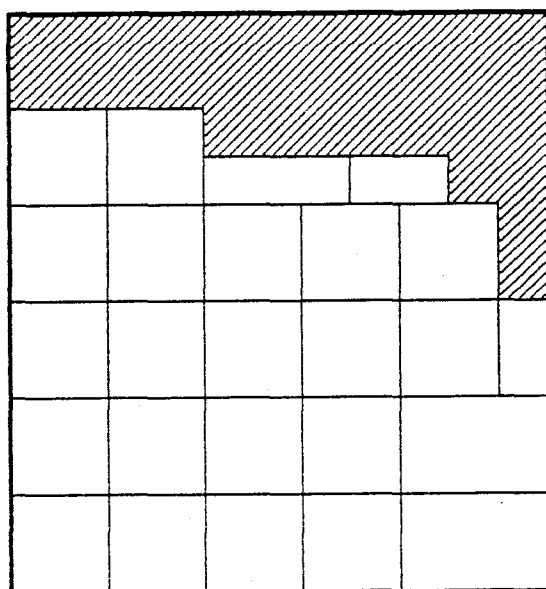


図 3. 6 アルゴリズム 3. 1 によるパッキング例 6
（手続き odd）

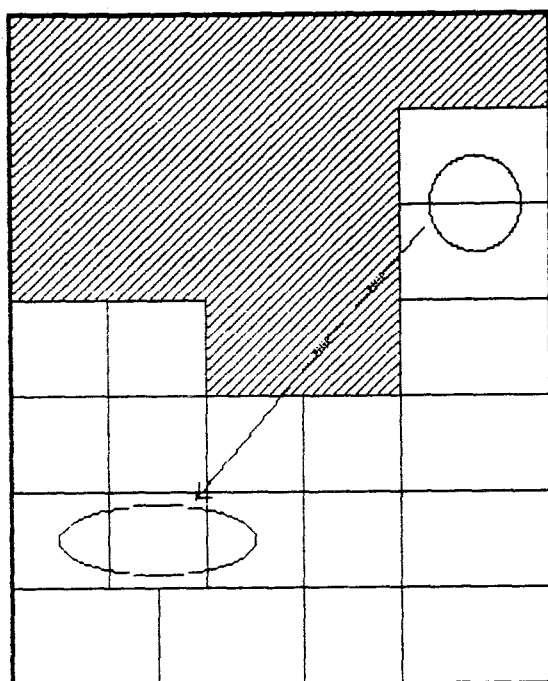


図 3. 7 アルゴリズム 3. 1 によるパッキング例 7
(手続き exchange)

後は，幅 W' の一番上のレベルを除いてタイトであり， p_{32} ができる限り p_{22} と交換されている．

ここで， p_{22} が十分あり， p_{32} が手続き exchange でできる限り交換されているならば，手続き repack で残った p_{32} は 1 個以下である．1 個以下の場合も (1) の場合と同様に，手続き repack で一番上のタイトでない (唯一の) レベルの続きに p_{32} を BL で詰めれば近似度は保たれる．また，手続き repack で p_{32} が 2 個以上ある場合も (1) の場合と同様に， p_{32} ， p_{22} の順に BL で詰めれば近似度は保たれる．

以上のことから， $W = \text{奇数}$ の場合にこのアルゴリズムで詰めると，近似度は $H_A \leq H_{opt} + 1$ となる（図 3. 8）．

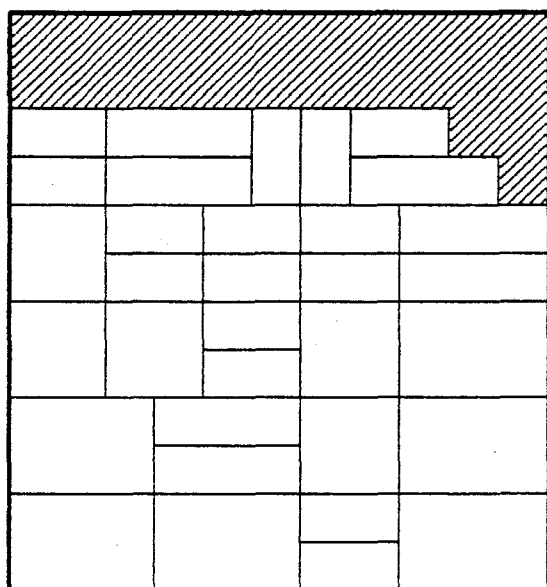


図 3. 8 アルゴリズム 3. 1 によるパッキング例 8
（手続き odd）

（3） アルゴリズムの時間計算量

〔定理 2. 1〕の証明の場合と同様に，このアルゴリズムは，1 つのピースをただか定数回しか詰め直すことを行わないので， $O(n)$ 回の BL パッキングで行うことができる。□

3. 3 近似解法の計算機実験

本節では，ピースの種類を $w \times 1$ および $w \times 2$ ($w \geq 5$) の部分矩形に限定した場合について，前章のピースの種類が 4×1 の部分矩形である場合の厳密解法を利用した近似解法を提案し，その良さ

を既存の近似解法であるレベル・アルゴリズム^[6]との比較実験を通して評価している。

3.3.1 ピースの種類が $w \times 1$ および $w \times 2$ の部分矩形である場合

まず、ピースの種類を $w \times 1$ ($w \geq 5$) の部分矩形に限定した場合を問題にする。この場合には、ピースを w が 5 以上のものと 4 以下のものに大別して処理する手法が考えられる。アルゴリズム 3. 2 はこの考えに基づいて提案したものである。

アルゴリズム 3. 2

- step 1 幅 w から幅 5 までのピースを幅の大きなピースから順に BL で詰める；
- step 2 step 1 で詰められた高さまで、幅 4 から幅 1 までのピースをその順に BL で詰める；
- step 3 その上に残った幅 4 から幅 1 までのピースをアルゴリズム 2. 1 で詰める；

次に、ピースの種類を $w \times 2$ ($w \geq 5$) の部分矩形に限定した場合を問題にする。この場合には、高さ 1 のピースを縦に 2 つ重ねてすべて高さ 2 のピースとして扱うものと、高さ 1 のまま残るたかだかそれぞれ 1 個のピースに大別して処理する手法が考えられる。アルゴリズム 3. 3 は、この考えに基づいて、高さが 2 のピースに対してアルゴリズム 3. 2 を適用するとともに、高さ 1 のピースを常に BL で詰めるのではなく、より良い解にするための交換手続きを工夫したものである。

アルゴリズム 3. 3

- step 1 幅 2 以上で高さ 1 のピースについて、同じ幅のピースを

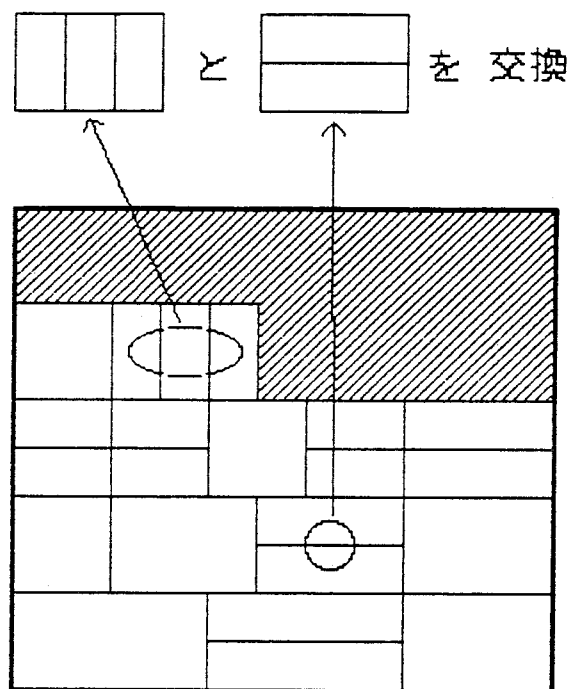


図 3 . 9 アルゴリズム 3 . 3 の step 3 の操作

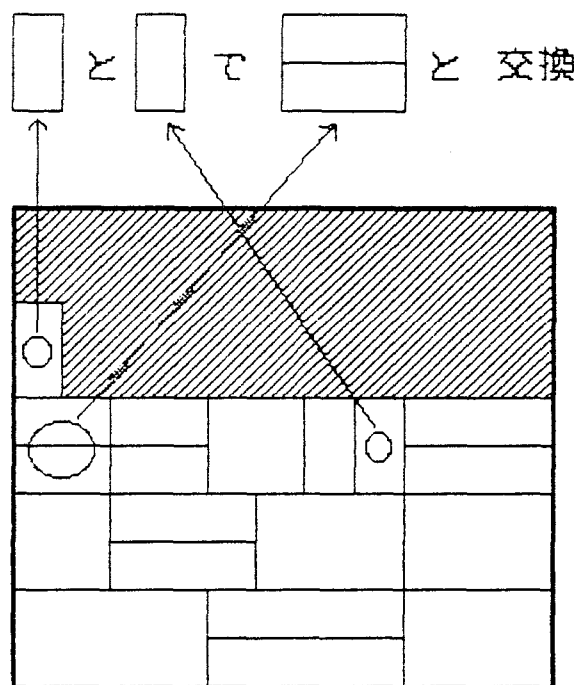


図 3. 1 0 アルゴリズム 3. 3 の step 4 の操作

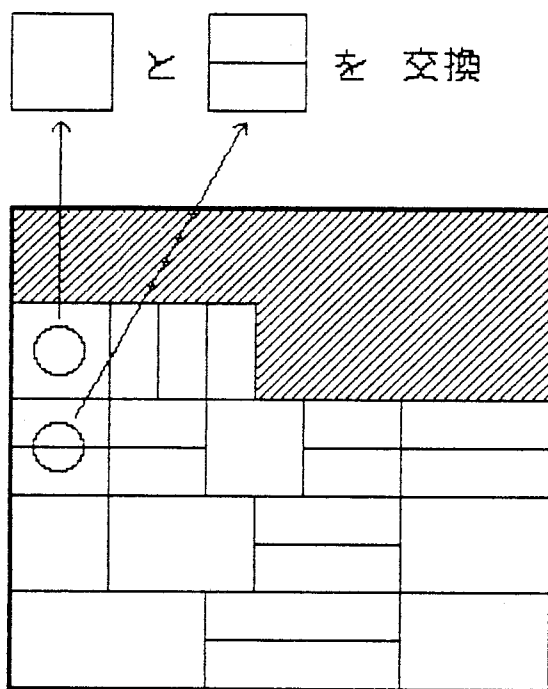


図 3 . 1 1 アルゴリズム 3 . 3 の step 5 の操作

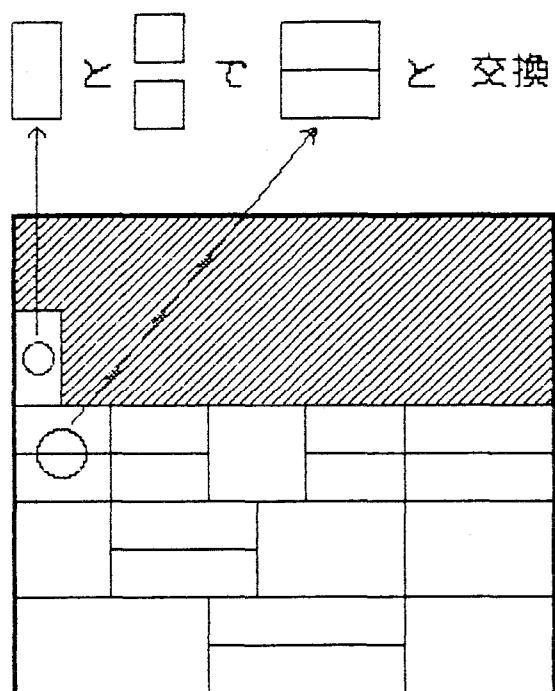


図 3 . 1 2 アルゴリズム 3 . 3 の step 6 の操作

- 縦に2つ並べて高さ2のピースとする； ただし、ピースが1つ余る場合はそのまま残しておく；
- step2 高さ2のピースを前述のアルゴリズム3.2で詰める；
- step3 タイトでない領域の m 個 ($m \geq 2$) の p_{12} を高さ1のピースを重ねて作られた p_{m2} と交換する (図3.9)；
- step4 タイトでない領域の p_{12} が1つのときは、タイトな領域から p_{12} を1つ取ってきて、それらのピースを高さ1のピースを重ねて作られた p_{22} と交換する (図3.10)；
- step5 タイトでない領域の高さ2のピースで作られたピースと、タイトな領域の高さ1のピースを重ねて作られた同一幅のピースを交換する (図3.11)；
- step6 タイトでない領域に p_{12} がまだ1つだけ残っている場合は、 p_{11} 2個より p_{12} を作り、これらの p_{12} 2個を高さ1のピースを重ねて作られた p_{22} と交換する (図3.12)；
- step7 残っている高さ1のピースと交換された高さ1のピースを幅の大きなピースから順にBLで詰める；
- step8 p_{11} をBLで詰める；

3.3.2 計算機実験による評価

前項で提案した近似解法を、計算機実験によって既存の近似解法であるFFDH-レベル・アルゴリズム^[6]との比較で評価する。比較基準として、FFDH-レベル・アルゴリズムを取り上げたのは、このアルゴリズムが最悪の場合の解の振舞いの点から考えて、最も良い近似解法の1つであることが知られているからである。

ふつう、近似解法での評価基準は、“近似解／最適解の平均値”である近似度を用いる。しかし、本節では、“すべての近似解の中で最適解と一致した解の割合(%)”を最適率と呼び、これを評価

基準に用いる。この理由は、アルゴリズム 3. 2, アルゴリズム 3. 3 で求めた近似解がすべて最適解 + 1 で収まり、近似度に差異が少なかったからである。

計算機実験にあたっては、問題例を (1) 矩形分割法 (RDG), (2) 乱数ピース発生法 (RPG) によって生成するものとし、ビンの幅を 10 から 40 の範囲で、ピース数を 40 から 400 の範囲で変更して試行し、各試行に対しては 100 個の問題例を生成している。なお、RPG 法により生成される問題例に対しては、ピースの面積の総和から下界を求め、それを最適解の代わりに用いる。

1) アルゴリズム 3. 2 の評価

アルゴリズム 3. 2 とレベル・アルゴリズムの実験による最適率を表 3. 1 に示す。ピースの幅の最大値が 5 の場合は、RDG も RPG も 16 試行 (1600 例) 行い、ピースの幅の最大値が 8 と 15 の場合は、いずれも、RDG も RPG も 8 試行 (800 例) 行った。アルゴリズム 3. 2 は問題例 6400 すべてに対して最適解を得ている。

表 3. 1 からわかるように、RPG より RDG で問題例を生成した方が解が悪い傾向にある。これは RDG で生成した最適解はすべてタイトな最適解であり、最適解の自由度が少ないためであると考えられる。また、表 3. 1 ではわからないが、各試行による最適率の差は大きく、ピース数が多くなればなるほど最適率が良くなる傾向にあった。

2) アルゴリズム 3. 3 の評価

アルゴリズム 3. 3 とレベル・アルゴリズムの実験による最適率を表 3. 2 に示す。ピースの幅の最大値が 5 の場合は、RDG は 20 試行 (2000 例), RPG では 16 試行 (1600 例) 行い、ピースの幅の最大値が 8 と 15 の場合は、いずれも、RDG も RPG も 8 試行 (800 例) 行った。アルゴリズム 3. 3 はいずれ

表 3. 1 アルゴリズム 3. 2 の実験結果 (最適率)

ピースの幅の 最大値 (w)	レベル・アルゴリズム		アルゴリズム 3. 2	
	R D G	R P G	R D G	R P G
5	69.9%	94.0%	100.0%	100.0%
8	24.6%	75.3%	100.0%	100.0%
15	15.4%	69.3%	100.0%	100.0%

表 3. 2 アルゴリズム 3. 3 の実験結果 (最適率)

ピースの幅の 最大値 (w)	レベル・アルゴリズム		アルゴリズム 3. 3	
	R D G	R P G	R D G	R P G
5	4.6%	46.1%	100.0%	100.0%
8	0.6%	19.8%	99.6%	98.9%
15	0.3%	12.3%	96.9%	97.3%

の場合にも 95% 以上の最適率を得ている。

アルゴリズム 3. 2 の場合と同様に, R P G より R D G で問題例を生成した方が解が悪い傾向にある。また, 表 3. 2 ではわからないが, アルゴリズム 3. 2 の場合と同様に, 各試行による最適率の差は大きく, ピース数が多くなればなるほど, 最適率は良くなる傾向にあった。また, ピースの幅の最大値 w が大きくなればなるほど,

すなわち，ピースの種類が多くなればなるほど，最適率は悪くなる傾向にあった．

3. 4 結言

本章では，ビンの幅，ピースの幅，高さを整数値に制約した2次元ビン・パッキング問題を取り上げ，その近似解法を考察した．まず，近似解法の理論的解析として，ピースの種類を 3×2 の部分矩形に限定した場合の近似解法を提案し，その解法が線形時間で最悪の場合でもたかだか高さが1しか悪くならない近似解が得られることを示した．次に，ピースの種類を $w \times 1$ および $w \times 2$ ($w \geq 5$)の部分矩形に限定した場合の近似解法を提案し，その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価した．実験結果から，提案した2つの近似解法は，計算時間も妥当であり，性能の点でも良い近似解法であると評価している．

第4章 ボロノイ図を応用した 真円度を求める解法

4.1 緒言

本章では，ボロノイ図を応用した真円度を求める多項式時間の厳密解法を提案し，その最適性と時間計算量を明らかにする．

真円度^[16]は，機械部品の精度として重要な幾何公差（形状公差）の1つであり，“円筒断面の輪郭を2つの同心円ではさんだとき，両円の間隔が最小となる両円の半径差”と定義される（図4.1）．

真円度を求める手法としては，これまでのところ，最小自乗法による解法，Min-Max法を適用した解法，シンプレックス法を用いた解法等が考えられているが，近似解法に過ぎないとか，厳密解法で

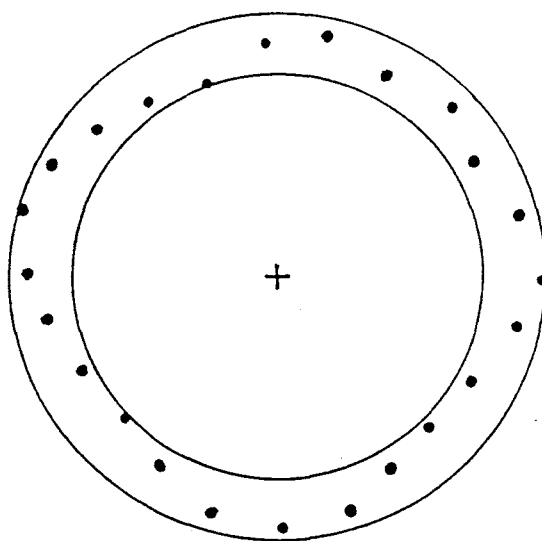


図4.1 真円度

あっても時間計算量に難点があるものであって，時間計算量が多項式時間である厳密解法は知られていない．

4. 2 最近点および最遠点のボロノイ図とそれらの構成算法

4.2.1 最近点および最遠点のボロノイ図

ボロノイ図（勢力圏図）^{[8][13]}とは，次のように定義されるものである．

“平面上の n 個の点 $p_i = (x_i, y_i)$ ($i=1, 2, \dots, n$) に対して，点 p_i の‘勢力圏’ $V(p_i)$ を

$$V(p_i) = \bigcap_{j=1}^n \{p = (x, y) \mid d(p_i, p) \leq d(p_j, p)\}$$

と定め， $V(p_i)$ ($i=1, 2, \dots, n$) による平面分割をボロノイ図と呼

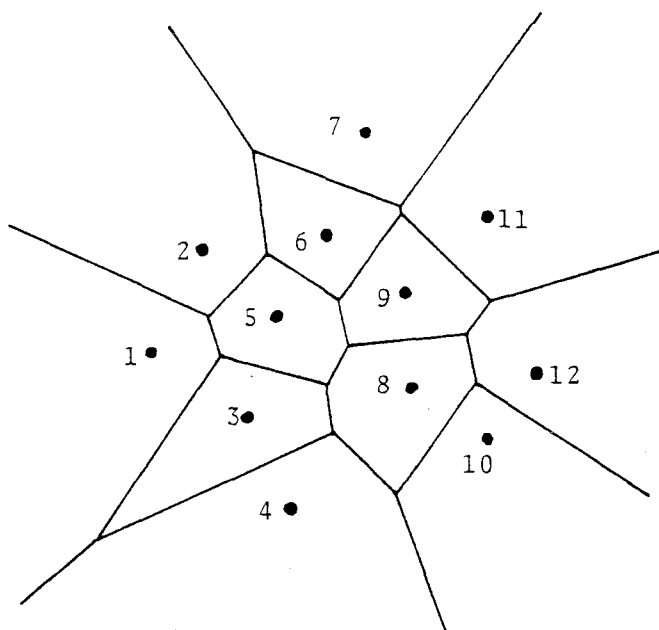


図 4. 2 最近点のボロノイ図

ぶ（図4.2）”。ここで、 $d(p_i, p)$ は2点 p_i, p のユークリッド距離，すなわち、 $d^2(p_i, p) = (x - x_i)^2 + (y - y_i)^2$ である。

ボロノイ図において，‘勢力圏’ $V(p_i)$ を点 p_i のボロノイ領域と呼び，これは凸多角形である。また，ボロノイ領域の頂点をボロノイ点，辺をボロノイ辺と呼ぶ。

以上は，最近点（勢力圏）を表すボロノイ図であるが，これに対して，最遠点（非勢力圏）を表すボロノイ図^[13]も考えられ，次のように定義される。

“平面上の n 個の点 $p_i = (x_i, y_i)$ ($i=1, 2, \dots, n$) に対して，点 p_i の‘非勢力圏’ $U(p_i)$ を

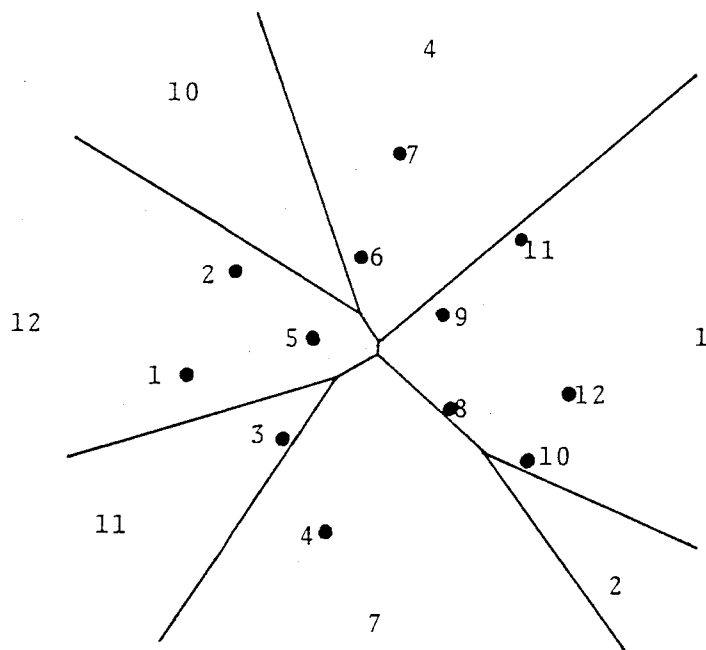


図4.3 最遠点のボロノイ図

$$U(p_i) = \bigcap_{j=1}^n \{p = (x, y) \mid d(p_i, p) \geq d(p_j, p)\}$$

と定め、 $U(p_i)$ ($i=1, 2, \dots, n$) による平面分割を最遠点のボロノイ図と呼ぶ (図 4. 3) ”.

最遠点のボロノイ図の場合はすべての点が領域を持つわけではなく、与えられた n 個の点の凸包上の点のみ領域を持つ。

4.2.2 最近点および最遠点のボロノイ図の構成算法

ボロノイ図の構成算法は、分割統治法を用いる手法が考えられている。最近点のボロノイ図の構成算法をアルゴリズム 4. 1 に示す [13] .

アルゴリズム 4. 1

- step1 n 個の点を x 座標の値に関してソートし、その順に $p_1, \dots, p_n$ とする；
- step2 これらの点を左側の点の集合 $L = \{p_1, \dots, p_{\lfloor n/2 \rfloor}\}$ と右側の点の集合 $R = \{p_{\lfloor n/2 \rfloor + 1}, \dots, p_n\}$ に二等分し、それぞれに対するボロノイ図 $V(L)$, $V(R)$ を再帰的に構成する；
- step3 点集合 L , R から等距離な点の集合の軌跡 (折れ線) を求め、その線より左にある $V(R)$, その線より右にある $V(L)$, それぞれのボロノイ点, ボロノイ辺を除くことによって、2つのボロノイ図 $V(L)$, $V(R)$ を併合する (図 4. 4) ；

step3 の併合は併合する図の点の数に比例する手間で行えるので、アルゴリズム 4. 1 の時間計算量は $O(n \log n)$ である [13] .

次に、最遠点のボロノイ図の構成にあたっては、アルゴリズム

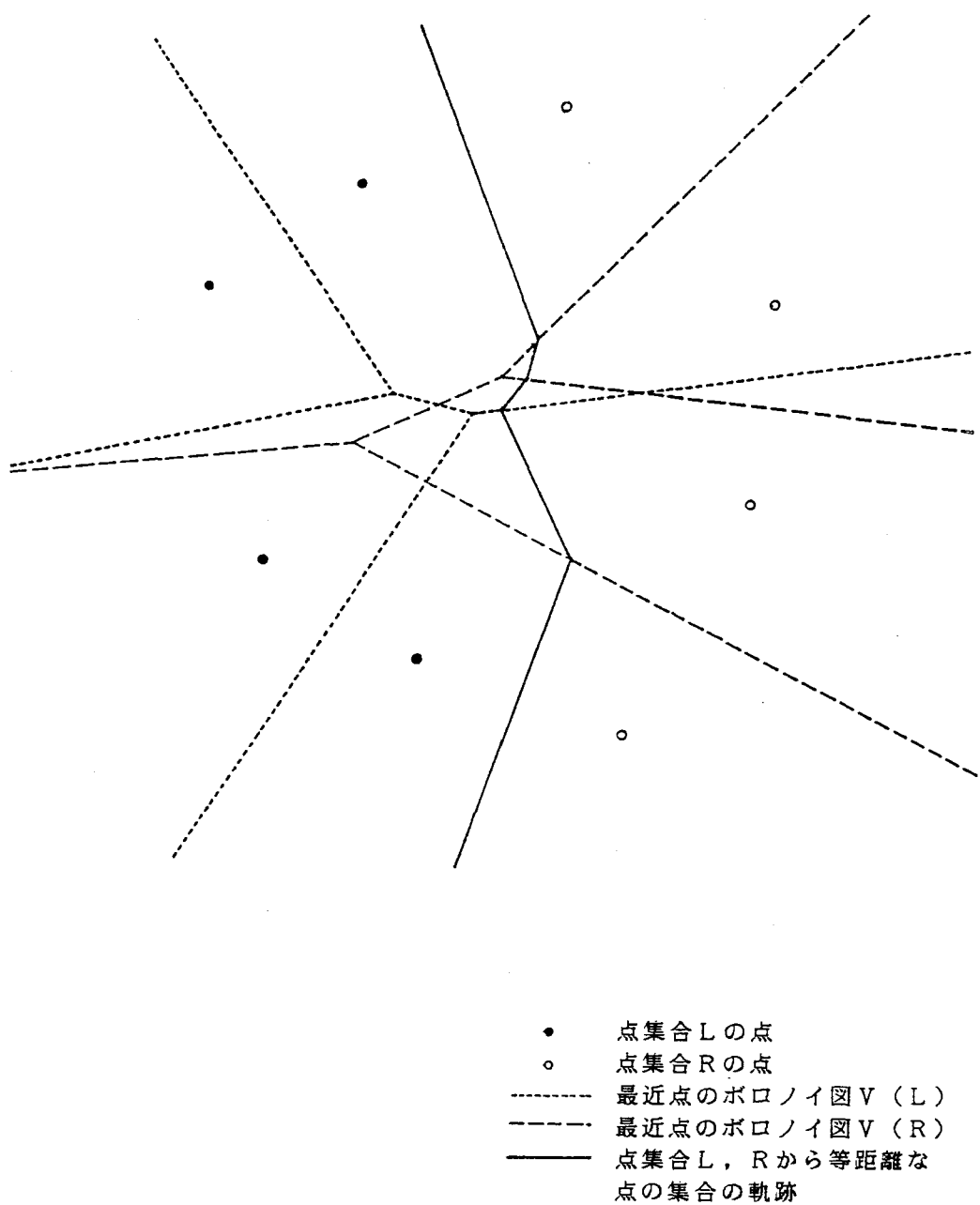


図4.4 最近点のボロノイ図の併合

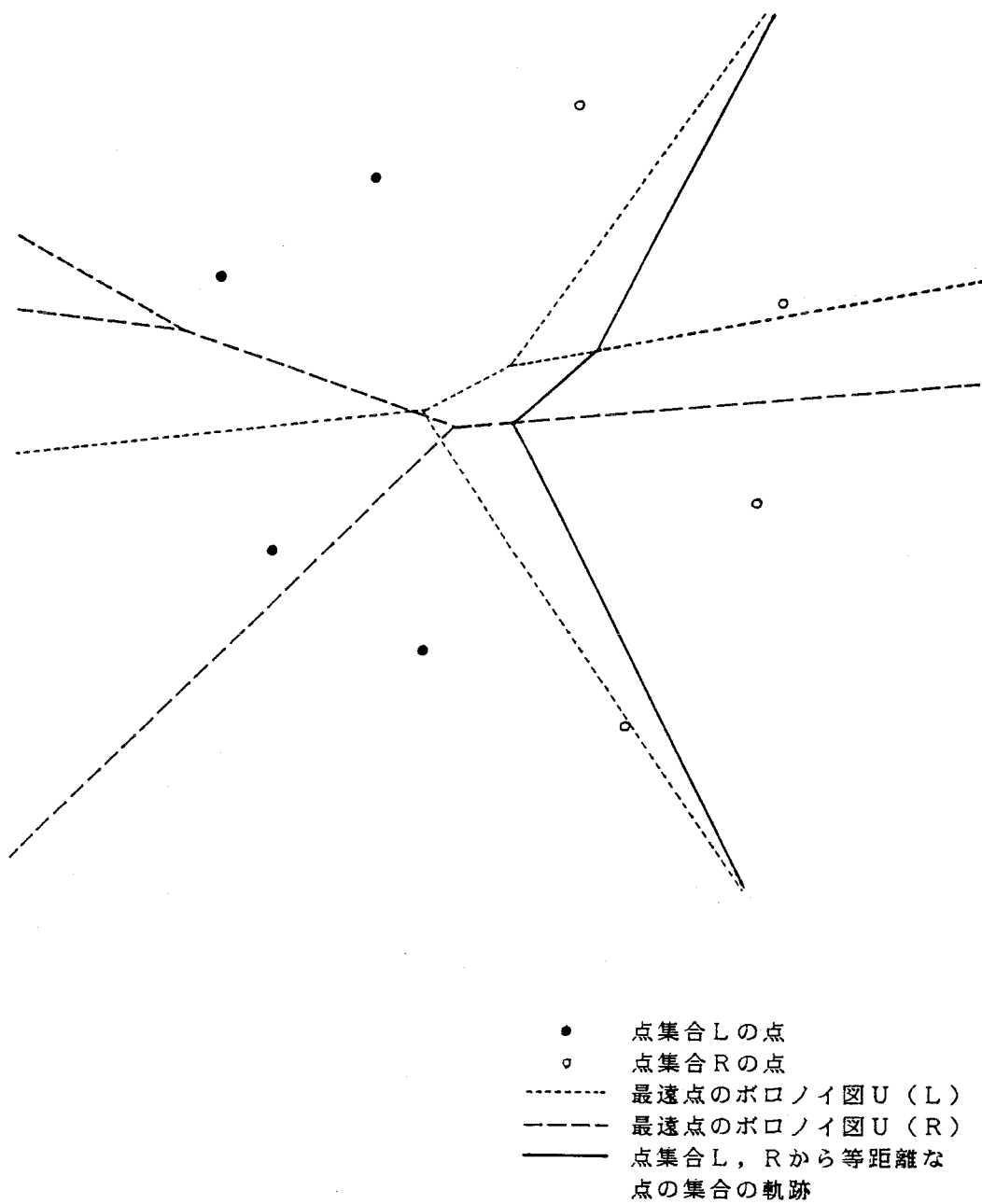


図4.5 最遠点のボロノイ図の併合

4. 1で等距離な点の集合の軌跡より遠い側を取り除いたのに対して、近い側を取り除けば済む。したがって、最遠点のボロノイ図の構成算法はこの点を除いてアルゴリズム4. 1と同じである。この算法をアルゴリズム4. 2に示す^[13]。

アルゴリズム4. 2

- step1 n 個の点を x 座標の値に関してソートし、その順に $p_1, \dots, p_n$ とする；
- step2 これらの点を左側の点の集合 $L = \{p_1, \dots, p_{\lfloor n/2 \rfloor}\}$ と右側の点の集合 $R = \{p_{\lfloor n/2 \rfloor + 1}, \dots, p_n\}$ に二等分し、それぞれに対するボロノイ図 $U(L)$, $U(R)$ を再帰的に構成する；
- step3 点集合 L , R から等距離な点の集合の軌跡（折れ線）を求め、その線より左にある $U(L)$ 、その線より右にある $U(R)$ 、それぞれのボロノイ点、ボロノイ辺を除くことによって、2つのボロノイ図 $U(L)$, $U(R)$ を併合する（図4. 5）；

アルゴリズム4. 2の時間計算量も同様に $O(n \log n)$ である^[13]。

4. 3 真円度を求める解法

円筒断面の輪郭が n 個の点で与えられているとする。このとき、真円度を決める同心円の中心はその点からの最も近い点（1点とは限らない）と最も遠い点（1点とは限らない）の距離の差が最小となる点である。ところで、 n 個の点についてのボロノイ図を考えると、最近点のボロノイ領域内の点はその領域に対応する点が最近点であり、最遠点のボロノイ領域内の点はその領域に対応する点が最

遠点になっている．ここで，最近点のボロノイ図と最遠点のボロノイ図の結び（Union）をとると，平面はたかだか $O(n^2)$ 個の領域に分割され，各領域内の点は最近点と最遠点が唯一に決められている．

なお，最近点と最遠点のボロノイ領域はいずれも凸領域であり，2つの凸領域の交わり（Intersection）は凸領域であることから，2つのボロノイ図の結びとして作られる各領域は凸領域であることに注意しておく．

以上のことから，真円度を決める同心円の中心は，与えられた n 個の点に対する2つのボロノイ図の結びの領域について，その領域内の点で最近点および最遠点への距離の差を最小とする点を見い出せばよいことになる．すなわち，結びの領域 r 内の点を $p(x_p, y_p)$ ，その領域に対する最近点および最遠点をそれぞれ $v(x_v, y_v)$ ， $u(x_u, y_u)$ とすれば，問題は領域 r 内で距離の差の関

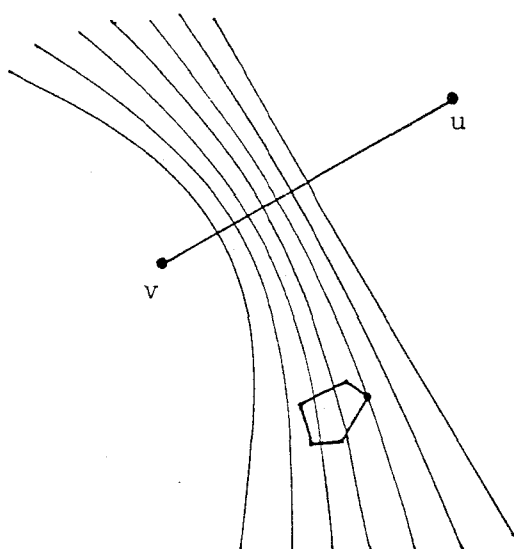


図 4. 6 距離の差の関数と凸領域

数

$$f = \sqrt{(x_u - x_p)^2 + (y_u - y_p)^2} \\ - \sqrt{(x_v - x_p)^2 + (y_v - y_p)^2}$$

を最小とすると定式化できる。

関数 f は双曲線となるので，結びの領域側（最近点側）からみると凹関数であり，結びの領域は先に指摘したように凸領域であるので，最適解は必ず結びの領域の頂点にある（図 4. 6）．したがって，問題の最適解の探索は，2つのボロノイ図の結びについて，最近点のボロノイ点，最遠点のボロノイ点，および，最近点のボロノイ辺と最遠点のボロノイ辺との交点のみを調べることで済む．

上述の論拠に基づいて真円度を求める解法としてアルゴリズム 4. 3 を提案する．

アルゴリズム 4. 3

- step1 アルゴリズム 4. 1 を行い，最近点のボロノイ図を構成する；
- step2 アルゴリズム 4. 2 を行い，最遠点のボロノイ図を構成する；
- step3 step1, 2 で求めた 2 つのボロノイ図の結び（Union）をとる（図 4. 7）；
- step4 2 つのボロノイ図の結びの図の各点（最近点のボロノイ点，最遠点のボロノイ点，最近点のボロノイ辺と最遠点のボロノイ辺との交点）に対して，その点からの最近点と最遠点の距離の差を求める；
- step5 最近点と最遠点の距離の差が最小となる点とその値を求める； 最小となる距離の差が真円度であり，最小となる点が真円度の同心円の中心である（図 4. 8）；

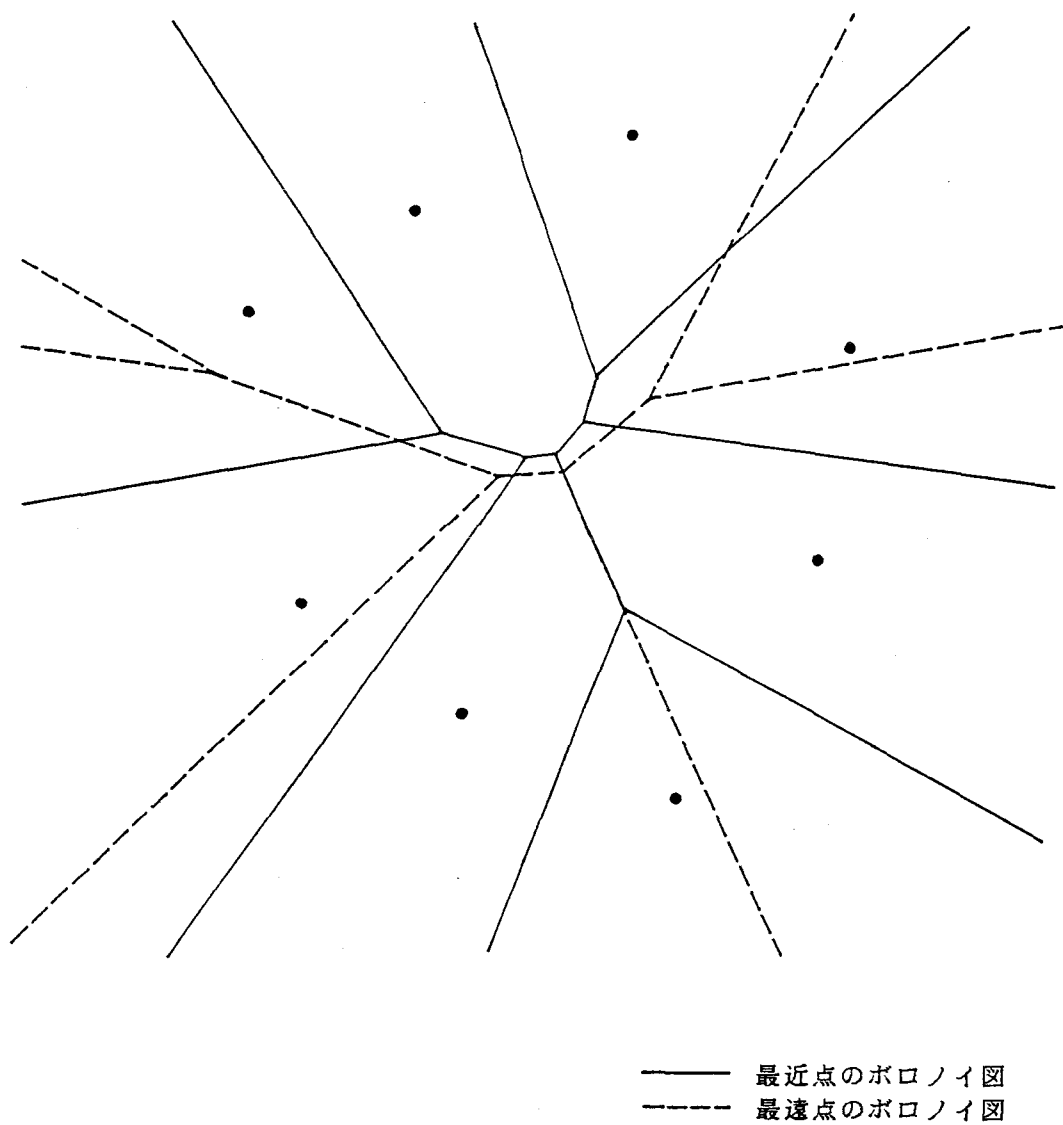
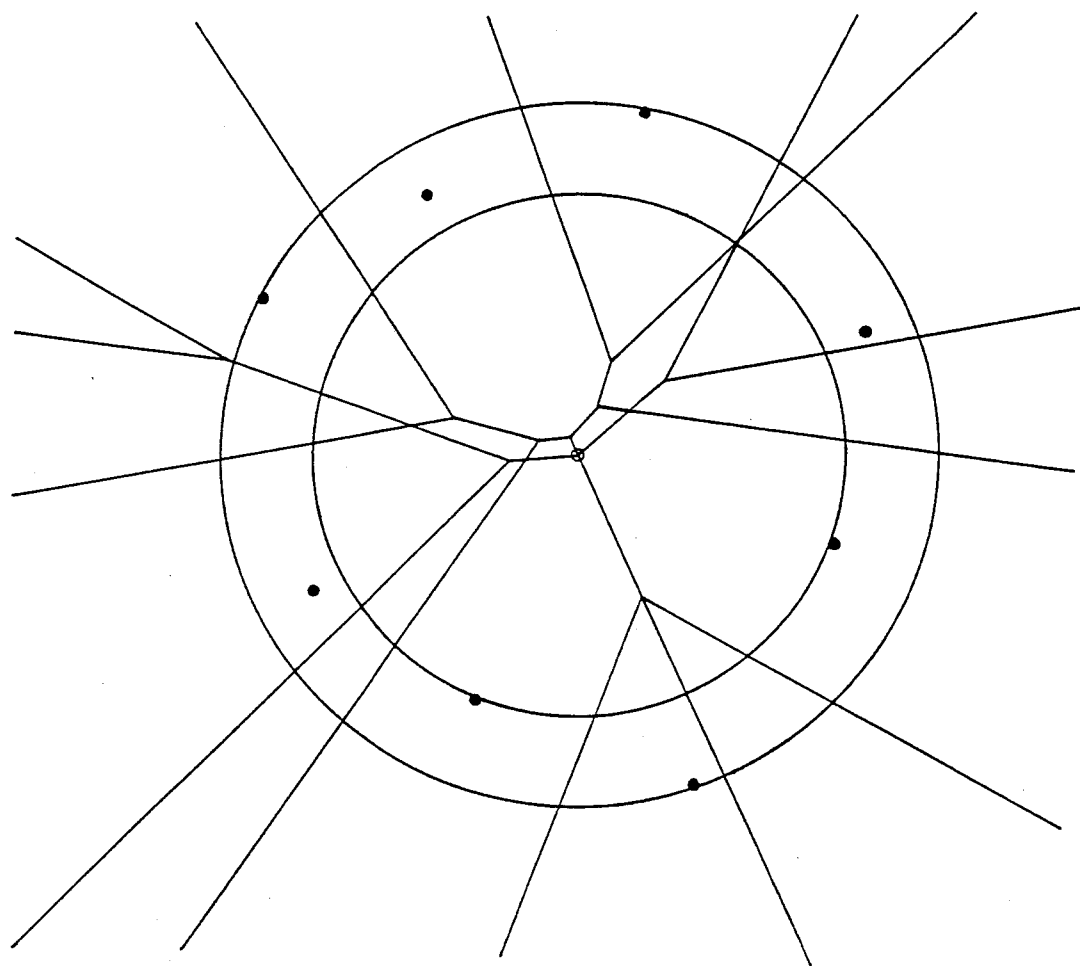


図4.7 最近点と最遠点のボロノイ図の結び



○ 真円度の同心円の中心

図 4 . 8 真円度の同心円

アルゴリズム 4. 3 の時間計算量

アルゴリズム 4. 3 における step 1, step 2 の最近点と最遠点のボロノイ図の構成は, 前節で示したように $O(n \log n)$ の時間計算量でできる. 2 つのボロノイ図は平面グラフであり, 頂点の数, 辺の数, 領域の数はすべて $O(n)$ である. したがって, step 3 で 2 つのボロノイ図の結びは $O(n^2)$ の手間で作ることができ, できあがった結びの図の頂点, 辺, 領域の数はたかだか $O(n^2)$ である. step 4 では, たかだか $O(n^2)$ 個の各点に対してその点からの最近点と最遠点の距離の差を求めるが, これは結び (Union) をとるときに各点の最近点と最遠点を 1 つずつ記憶しておくことによって各点に対して $O(1)$ の手間で行える. step 5 は, step 4 で計算された各点についての最近点と最遠点の距離の差のうちで, 最小となる点を見出す手続きであり, これに要する手間はたかだか $O(n^2)$ である.

以上のことから, アルゴリズム 4. 3 全体の時間計算量は, $O(n^2)$ ということになる.

4. 4 結言

本章では, 機械部品の精度にとって重要な因子の 1 つである真円度を求める解法として, 最近点のボロノイ図と最遠点のボロノイ図の結び (Union) をとるといふ考えに基づいた解法を提案し, その時間計算量が $O(n^2)$ であることを明らかにした.

本解法は, 計算幾何学の中心課題の 1 つであるボロノイ図の 1 つの応用を指摘するだけでなく, 真円度の決定問題に対して実用的にも有力な解法になるものと評価している.

また, 本問題を 3 次元に拡張すると, 真円度は真球度となり, 真球度も 3 次元のボロノイ図を構成することによって, 同様に多項式時間で求めることができる.

第5章 結論

本論文では、コンピュータ・サイエンスの中で急速に発展してきた計算幾何学の分野に属する2つの問題、2次元ビン・パッキング問題と真円度問題を取り上げた。

2次元ビン・パッキング問題については、ビンの幅、ピースの幅、高さをすべて整数値とし、ピースの種類を $w \times h$ の部分矩形にするという制約を加えた場合について、この問題の扱いやすさ (tractability) を考察し、次の成果を得た。

- (1) ピースの種類を 4×1 、 1×4 、 2×2 の部分矩形に限定した場合の解法を考案し、それらの解法が線形時間で最適解が得られるものであることを明らかにした。この結果は、2次元ビン・パッキング問題に対して線形時間の解法が存在する限界を明らかにしたものと考えている。
- (2) ピースの種類を 3×2 の部分矩形に限定した場合の解法を考案し、この解法が線形時間で最悪の場合でもたかだか高さが1しか悪くならない近似解が得られるものであることを示した。
- (3) ピースの種類を $w \times 1$ および $w \times 2$ ($w \geq 5$) の部分矩形に限定した場合の近似解法を考案し、その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価した。その結果、考案した2つの近似解法が、計算時間も妥当であり、性能の点でも既存の解法よりも良い近似解法であることを示した。

真円度問題については、計算幾何学の中心課題の1つであるボロノイ図を利用することによって、機械部品の精度として重要な幾何公差の1つである真円度を求める多項式時間の厳密解法が存在することを指摘し、その最適性と時間計算量を明らかにした。この解法

は、最近点のボロノイ図（勢力圏図）と最遠点のボロノイ図（非勢力圏図）の結び（Union）をとるという考えに基づいたものであり、その時間計算量は $O(n^2)$ である。

謝 辞

本研究の全過程を通じて、終始懇切丁寧な御指導、御激励を戴いた大阪大学工学部通信工学科中西義郎教授に衷心より感謝の意を表する。

大阪大学大学院在学中に、通信工学一般に関して御指導、御教示を戴いた大阪大学工学部通信工学科滑川敏彦名誉教授、熊谷信昭教授（現、大阪大学総長）、手塚慶一教授、倉藺貞夫教授に深謝する。

本研究を遂行するにあたり、終始直接の御指導、御助言を戴いた大阪大学工学部通信工学科中野秀男助手に心より感謝の意を表する。

日頃より、貴重な御意見、御討論を戴いた神戸大学工学部電気工学科岡田博美助教授、琉球大学工学部電子情報工学科川口剛助手、カノーブス電子株式会社稲村浩氏に深く感謝する。

真円度問題に関して、有益な御意見を戴いた新日本電工株式会社橋本良夫氏、株式会社東京精密真田友宏氏に感謝する。

日頃から、様々な面で御協力戴いた大阪大学工学部通信工学科浜富士雄技官を始めとする中西研究室の諸兄に御礼申し上げる。

参考文献

- [1] A.V.Aho, J.E.Hopcroft and J.D.Ullman : "Data Structures and Algorithms", Addison-Wesley (1982).
- [2] 浅野孝夫 : " 計算幾何学とその応用" , 情報処理, vol.25, no.3, pp.208-221 (Mar. 1984).
- [3] B.S.Baker, E.G.Coffman Jr. and R.L.Rivest : "Orthogonal Packings in Two Dimensions", SIAM J. Comput., vol.9, no.4, pp.846-855 (Nov. 1980).
- [4] K.R.Baker : "Introduction to Sequencing and Scheduling", John Wiley & Sons. Inc. (1974).
- [5] E.G.Coffman Jr., M.R.Garey and D.S.Johnson : "Approximation Algorithms for Bin-Packing : An Updated Survey", Algorithm Design for Computer System Design (edited by G.Ausiello, M.Lucertini and P.Serafini), Springer-Verlag, pp.49-106 (1984).
- [6] E.G.Coffman Jr., M.R.Garey, D.S.Johnson and R.E.Tarjan : "Performance Bounds for Level Oriented Two-Dimensional Packing Algorithms", SIAM J. Comput., vol.9, no.4, pp.808-826 (Nov. 1980).
- [7] M.R.Garey and D.S.Johnson : "Computers and Intractability : A Guide to the Theory of NP-Completeness", W.H.Freeman and Company (1979).
- [8] P.J.Green and R.Sibson : "Computing Dirichlet tessellations in the plane", The Computer Journal, vol.21, no.2, pp.168-173 (May 1978).
- [9] 伊理正夫監修 / 腰塚武志編集 : " 計算幾何学と地理情報処理" , bit 別冊, 共立出版 (Sep. 1986).

- [10] D.S.Johnson, A.Demers, J.D.Ullman, M.R.Garey and R.L.Graham : "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms", SIAM J. Comput., vol.3, no.4, pp.299-325 (Dec. 1974).
- [11] D.T.Lee and F.P.Preparata : "Computational Geometry-A Survey", IEEE Trans. on Computers, vol.c-33, no.12, pp.1072-1101 (Dec. 1984).
- [12] F.P.Preparata and M.I.Shamos : "Computational Geometry : An Introduction", Springer-Verlag (1985).
- [13] M.I.Shamos and D.Hoey : "Closest-Point Problems", Proc. 16th IEEE Symp. Foundations of Computer Science, pp.151-162 (Oct. 1975).
- [14] D.D.K.D.B.Sleator : "A 2.5 Times Optimal Algorithm for Packing in Two Dimensions", Info. Proc. Letters, vol.10, no.1, pp.37-40 (Feb. 1980).
- [15] R.E.Tarjan : "Data Structures and Network Algorithms", Society for Industrial and Applied Mathematics (1983).
- [16] 塚田忠夫, 金田徹, 奥田謙造 : "最適化技法を用いた最小領域法真円度の評価法", 精密機械, vol.49, no.10, pp.1351-1357 (Oct. 1983).